OpenNebula/Reservoir Training, January 27-28

Brussels, Belgium

Session 1
Introduction, Installation and Configuration

Daniel Molina & Javier Fontán
dmolina/jfontan@opennebula.org

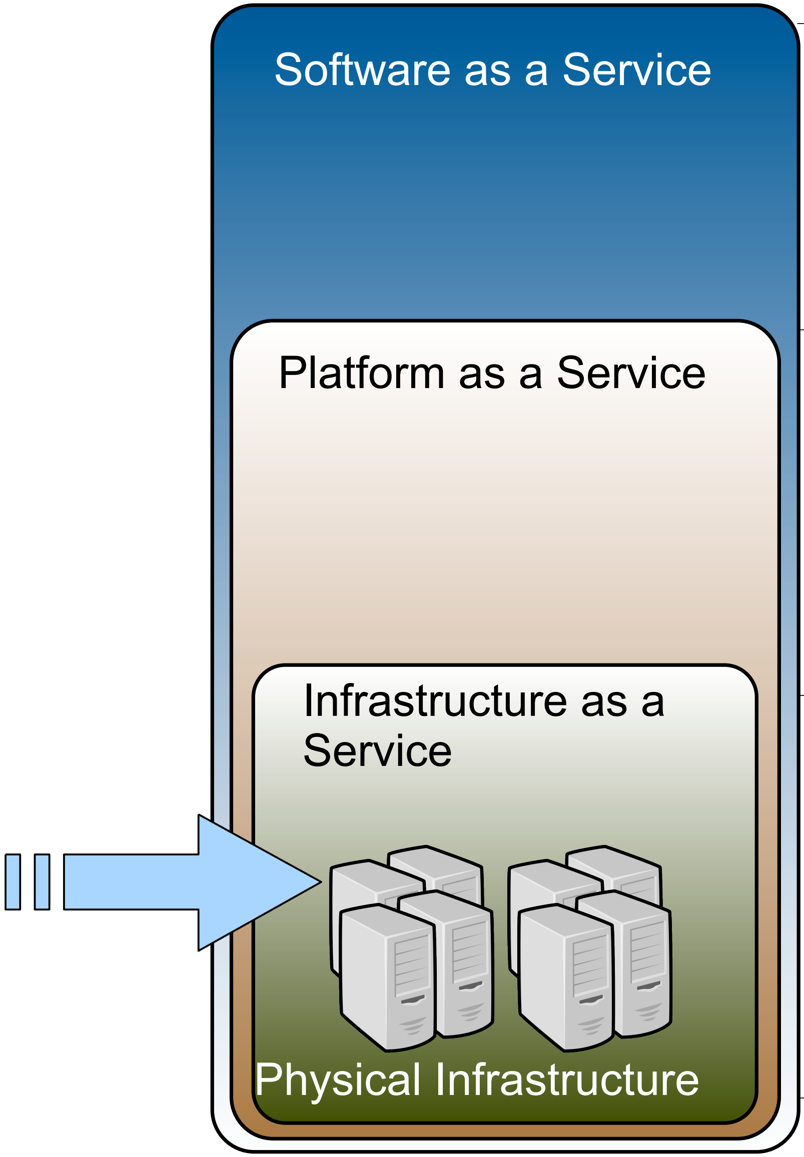
OpenNebula.org

RESERVOIR

SEVENTH FRAMEWORK PROGRAMME

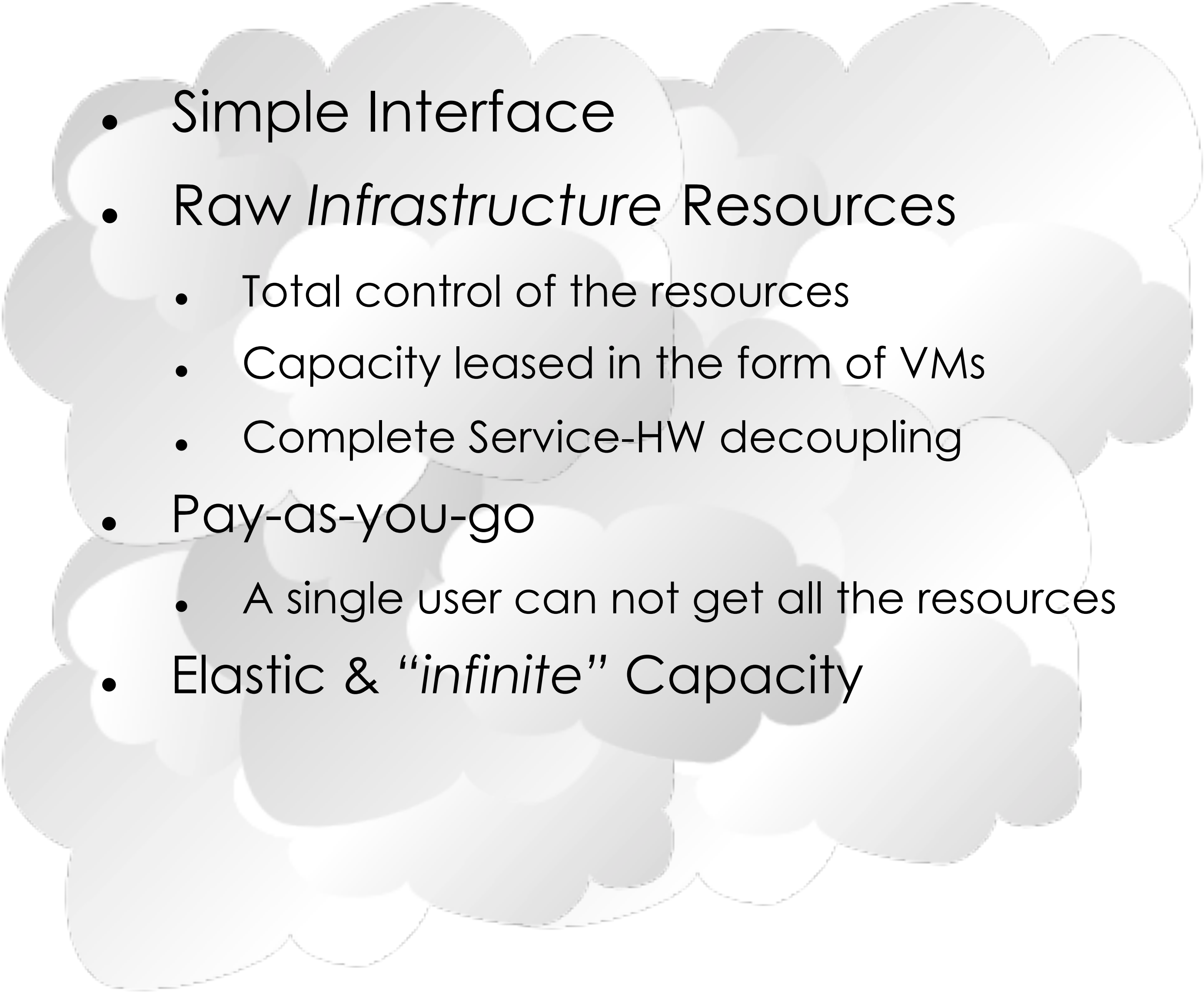
European Commission
Information Society
and Media

Cloud Computing in a Nutshell



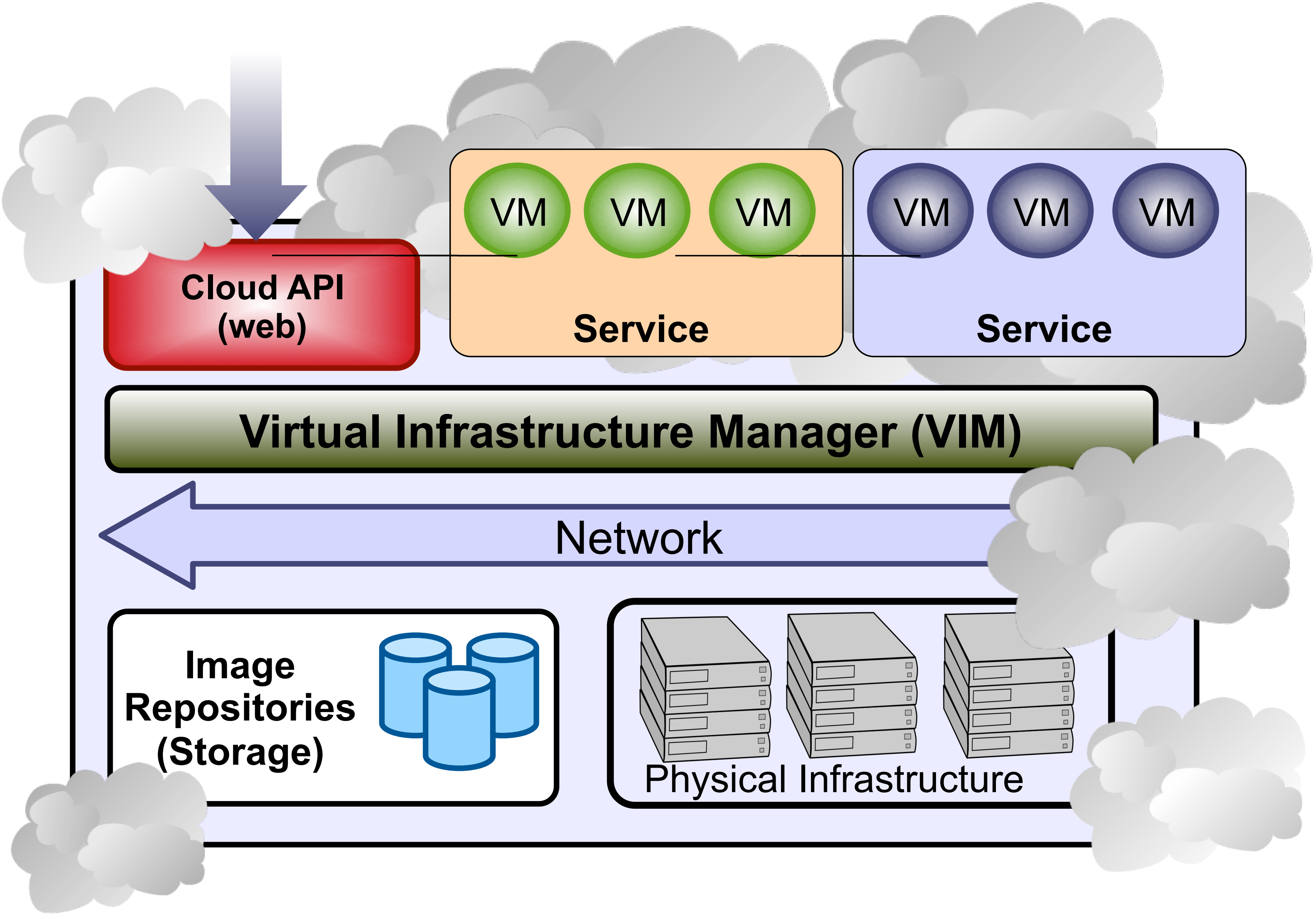
	What	Who
Software as a Service	On-demand access to any application	**End-user** (does not care about hw or sw) skype facebook
Platform as a Service	Platform for building and delivering web applications	**Developer** (no managing of the underlying hw & swlayers) Windows Azure force.com platform as a service
Infrastructure as a Service	Delivery of a *raw* computer infrastructure	**System Administrator** (complete management of the computer infrastructure) GOGRID flexiscale amazon web services

The IaaS Clouds a Four Point Check List



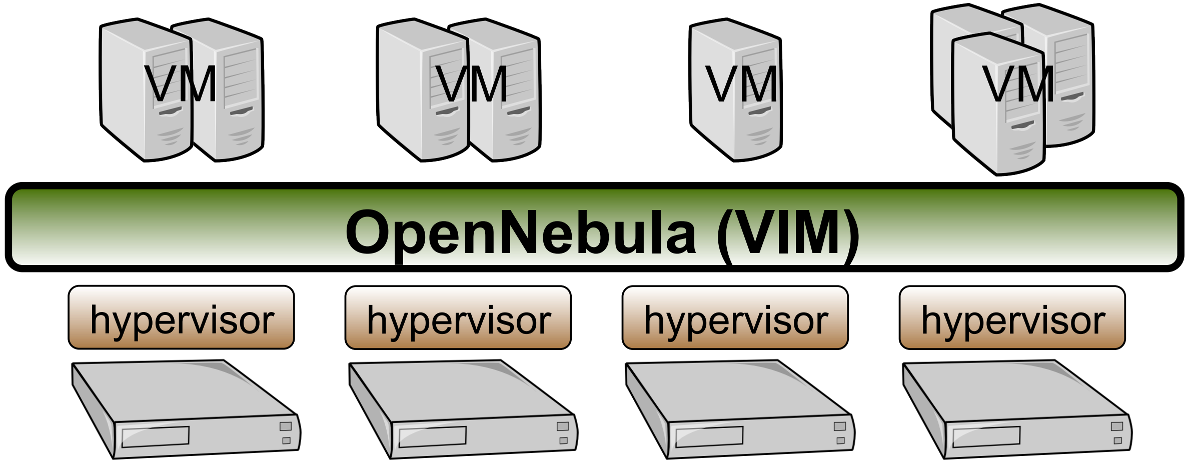
- Simple Interface
- Raw *Infrastructure* Resources
  - Total control of the resources
  - Capacity leased in the form of VMs
  - Complete Service-HW decoupling
- Pay-as-you-go
  - A single user can not get all the resources
- Elastic & “*infinite*” Capacity

The Anatomy of an IaaS Cloud



Why a Virtual Infrastructure Manager?

- VMs are great!!...but something more is needed
  - Where did/do I put my VM? (***scheduling & monitoring***)
  - How do I provision a new cluster node? (***clone & context***)
  - What MAC addresses are available? (***networking***)
- Provides a ***uniform view*** of the resource pool
- ***Life-cycle management*** and monitoring of VM
- The VIM ***integrates*** Image, Network and Virtualization

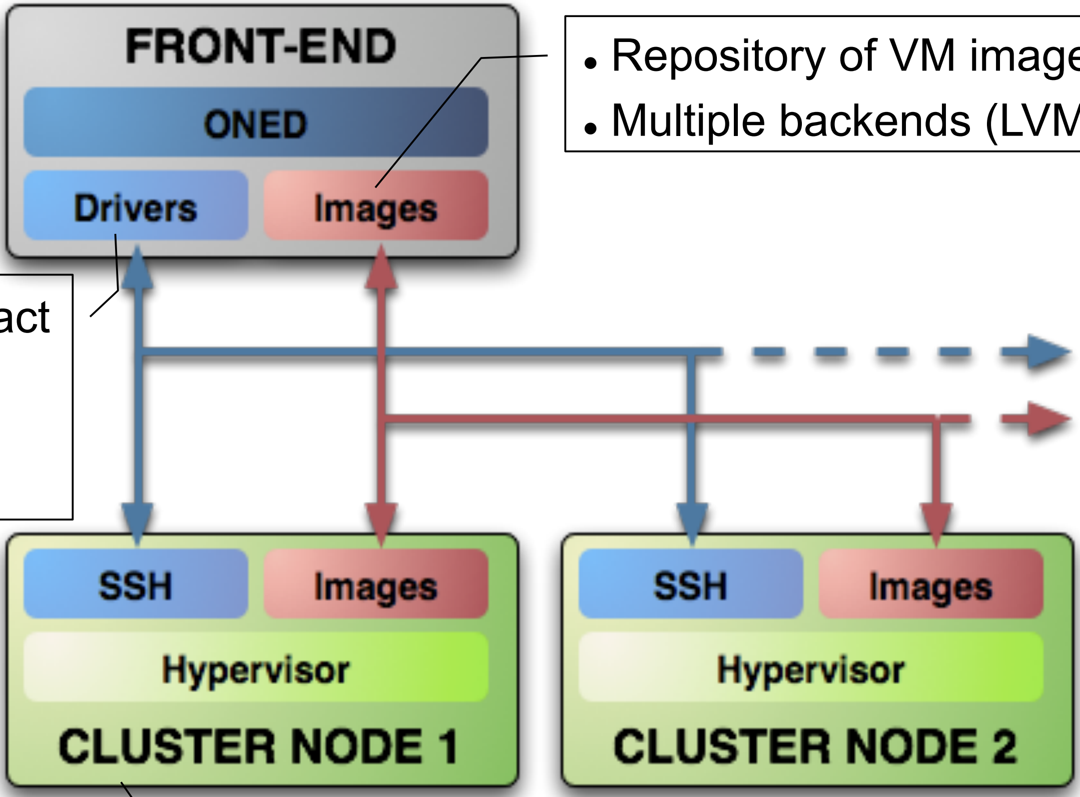


Planning the Installation: System Overview

- Executes the OpenNebula Services
- *Usually* acts as a classical cluster front-end

- Repository of VM images
- Multiple backends (LVM, iSCSI..)

- Modular components to interact with the cluster services
- *Types:* storage, monitoring, virtualization and network



- Provides physical resources to VMs
- *Must have* a hypervisor installed

⚠ The same host can be can be the front-end and a node

Planning the Installation: Working in the Front-End ...

- Choose your installation mode
  - system wide (/usr, /etc...)
  - *self-contained* (under $ONE_LOCATION)
- Install software dependencies.
  - Check the documentation for platform specific notes installation nodes

http://opennebula.org/documentation:rel2.0:notes

- Dependencies already installed in the Front-End and the Nodes

Planning the Installation: Working in the Front-End ...

- The Users of the private cloud:
  - oneadmin: Account to run the daemons, manage the system and do all the low-level operations (e.g. start VMs, move images...).
  - Regular users: create and manage their own VMs and networks. *Need to be defined in OpenNebula*
- Installation layout
  - We will use the /srv/cloud directory to place the OpenNebula software
  - /srv/cloud/one will hold the OpenNebula installation

```
# tree /srv
/srv
`-- cloud
    `-- one
        `-- SRC
```

⚠ The oneadmin account must be created system wide (i.e. front-end and all the nodes). You can use NIS, or a local account with the same ID's in all the hosts. Regular users do not need a UNIX account in the nodes, nor in the front-end.

Planning the Installation: Working in the Front-End ...

- Hands on...

```
Fe$ su -

fe# groupadd -g 9000 oneadmin

fe# mkdir /srv/cloud
fe# useradd -d /srv/cloud/one -g oneadmin -u 9000 -s /bin/bash -m
oneadmin

Create the file-system hierarchy with the oneadmin account

fe# su - oneadmin
fe$ id
uid=9000(oneadmin) gid=9000(oneadmin) groups=9000(oneadmin)

fe$ mkdir SRC

We will place the OpenNebula source code in SRC
```

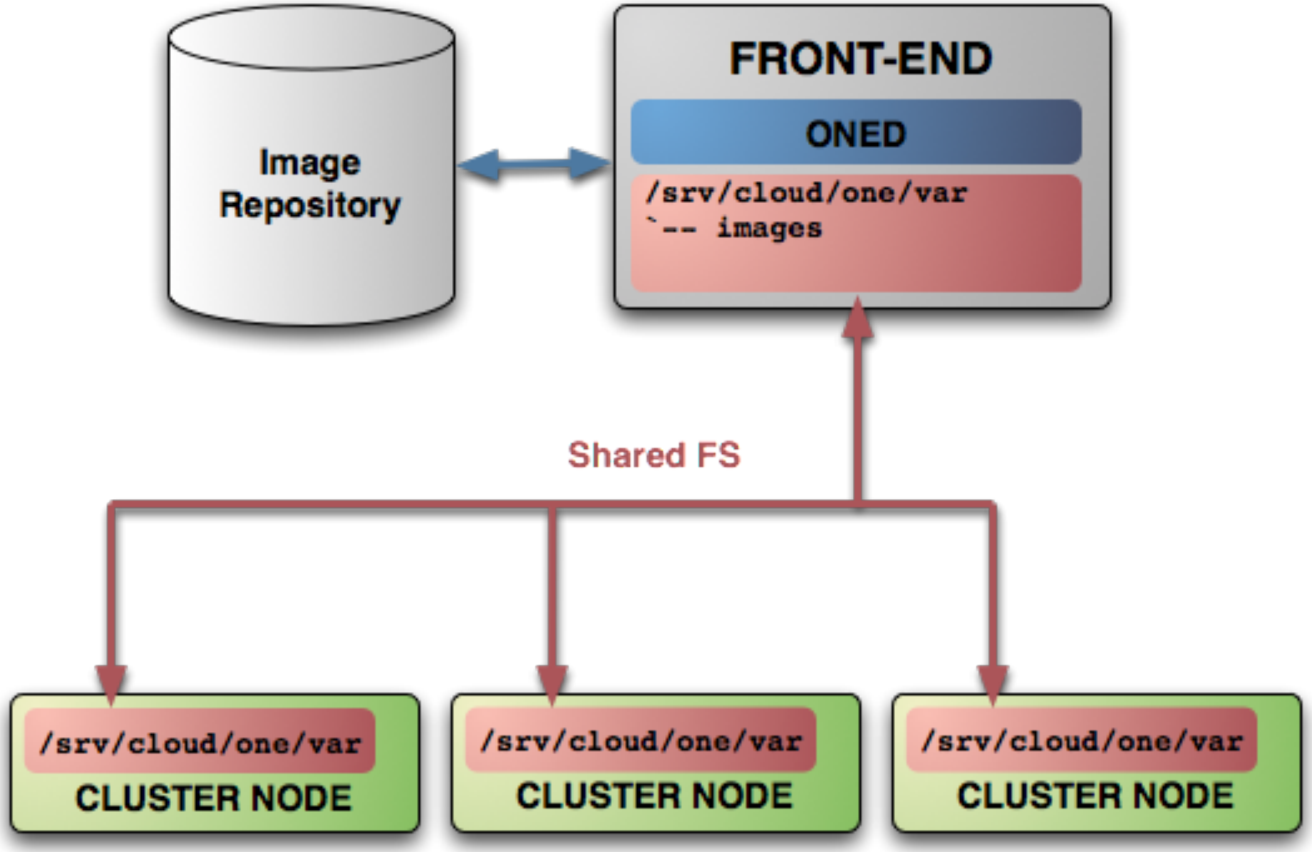
Planning the Installation: Working in the Front-End ...

- Preparing the storage for the private cloud...
  - ***Image Repository:*** Any storage medium for the VM images (usually a high performing SAN)
    - OpenNebula supports multiple back-ends (e.g. LVM for fast cloning)
    - The front-end must have access to the repository
  - ***VM Directory:*** The home of the VM in the cluster node
    - Stores checkpoints, description files and VM disks
    - Actual operations over the VM directory depend on the storage medium
    - Should be shared for live-migrations
    - You can go on without a shared FS and use the SSH back-end
    - Defaults to $ONE_LOCATION/var/$VM_ID

(!) ***Dimensioning the Storage...*** Example: A 64 core cluster will typically run around 80VMs, each VM will require an average of 10GB of disk space. So you will need ~800GB for /srv/cloud/one, you will also want to store 10-15 master images so ~200GB for image repository. A 1TB /srv/cloud will be enough for this example setup.

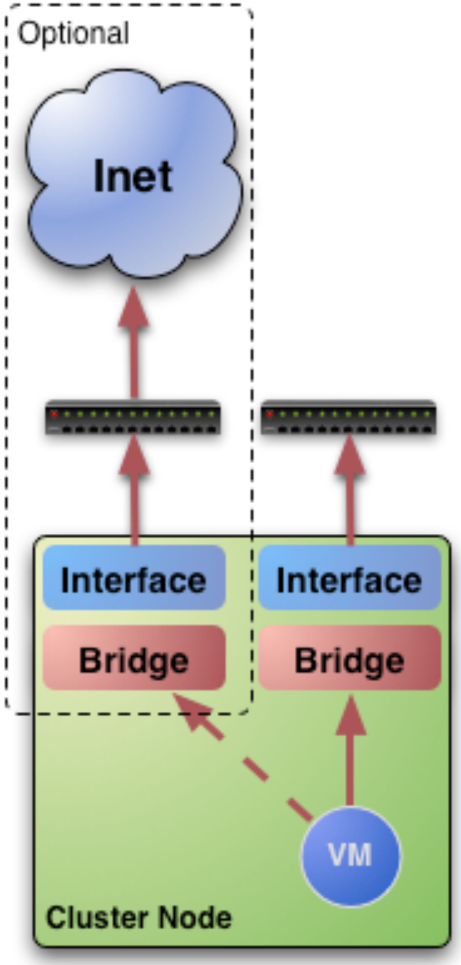
Planning the Installation: Working in the Front-End ...

- In this course we will use NFS to share the VM directories
- The Image Repository is /srv/cloud/one/var/images



Planning the Installation: Working in the Front-End ...

- Networking for the private cloud
  - OpenNebula management operations use ssh connections, it does not require a performing NIC
  - ***Image traffic,*** may require the movement of heavy files (VM images, checkpoints). Dedicated storage links may be a good idea
  - ***VM demands,*** consider the typical requirements of your VMs. Several NICs to support the VM traffic may be a good idea
  - OpenNebula relies on bridge networking for the VMs



Planning the Installation: Working in the Front-End ...

- Prepare NFS

```
Export /srv/cloud to your nodes
   - only need /srv/cloud/one/var
   - we also export $HOME of oneadmin for easy SSH key configuration

fe# vi /etc/exports
/srv/cloud cetic-nodeXX(rw,async,no_subtree_check,no_root_squash)

fe# /etc/init.d/nfs reload
```

Planning the Installation: Working in the Nodes ...

- Install software dependencies
  - We need SSH daemon running in the cluster nodes (check it!)
  - Runtime dependencies:
    - Ruby 1.8.x
- Users
  - Create the oneadmin account (**use same UID and GID**)

```
no# mkdir -p /srv/cloud
no# groupadd -g 9000 oneadmin
no# useradd -d /srv/cloud/one -g oneadmin -u 9000 -s /bin/bash oneadmin
```

  - Add oneadmin to sudoers

```
no# tail -1 /etc/sudoers
oneadmin ALL=(ALL) ALL, NOPASSWD: /usr/sbin/xm, /usr/sbin/xmtop
```

Planning the Installation: Working in the Nodes ...

- Storage
  - Recreate the installation layout and configure NFS to mount VM dirs

```
no# chown oneadmin:oneadmin /srv/cloud

no# vi /etc/fstab
frontend:/srv/cloud /srv/cloud nfs soft,intr,rsize=32768,wsize=32768,rw
0 0

no# mount /srv/cloud
```

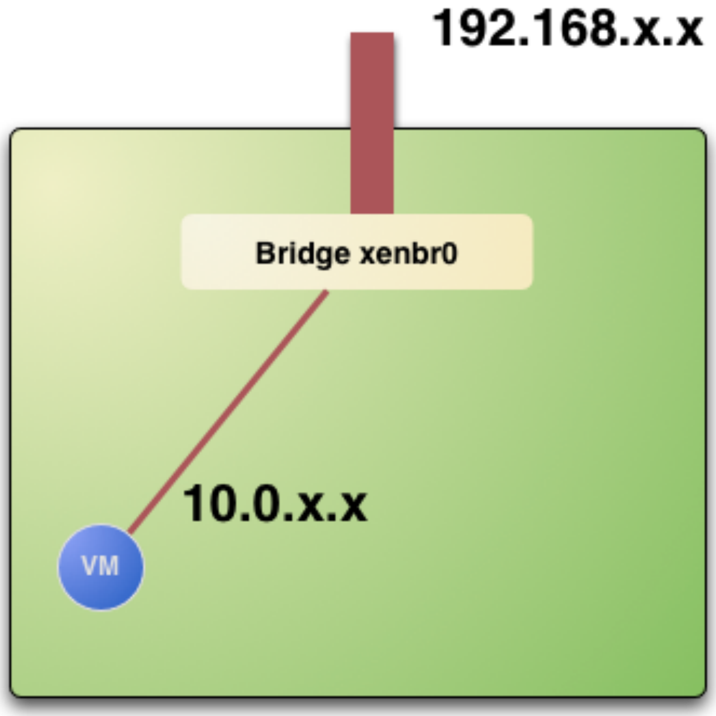
Planning the Installation: SSH Configuration

- Enable password-less SSH access to cluster nodes for the oneadmin account:

```
DO NOT PROTECT PRIVATE KEY WITH A PASSWORD
fe$ ssh-keygen
Generating public/private rsa key pair.
...
Enter passphrase (empty for no passphrase):
Enter same passphrase again:

fe$ cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys

Tell ssh client not to ask to add hosts to known_hosts (optional)

fe$ cat /srv/cloud/one/.ssh/config
Host *
    StrictHostKeyChecking no

TEST!
fe$ ssh localhost
fe$ ssh host01
```

Planning the Installation: The Hypervisor ...

- Installing the Hypervisor
  - OpenNebula supports KVM, Xen and Vmware (*even simultaneously)*. This course applies to KVM and Xen
  - Refer to the hypervisor documentation for additional (and better information) on setting up them.
  - In this course, we will use XEN.



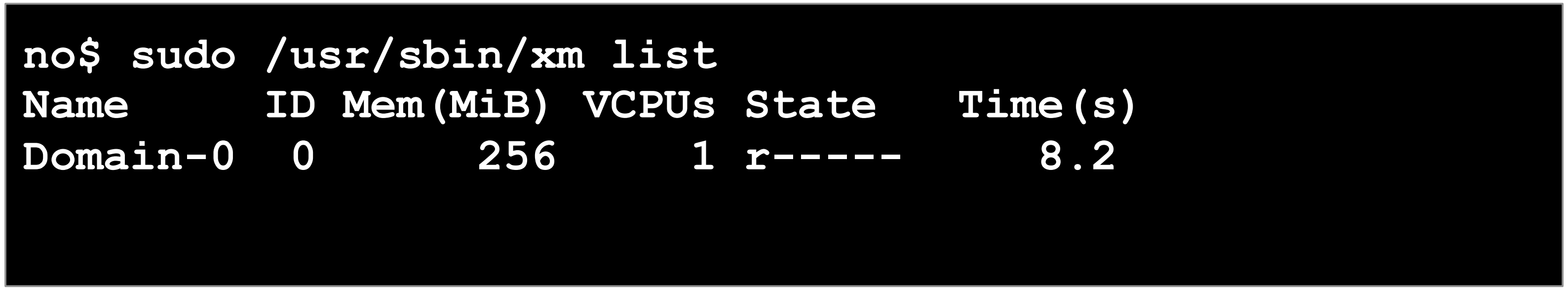
Planning the Installation: The Hypervisor ...

- The software bridge is essential for having different VMs in the same host with connectivity
- Let’s check the bridge in the hosts

```
no$ /usr/sbin/brctl show
Bridge name     bridge id               STP enabled     interfaces
virbr0          8000.000000000000       yes
xenbr0          8000.feffffffffff       no              peth0
                                                        vif0.0
```

Planning the Installation: The Hypervisor ...

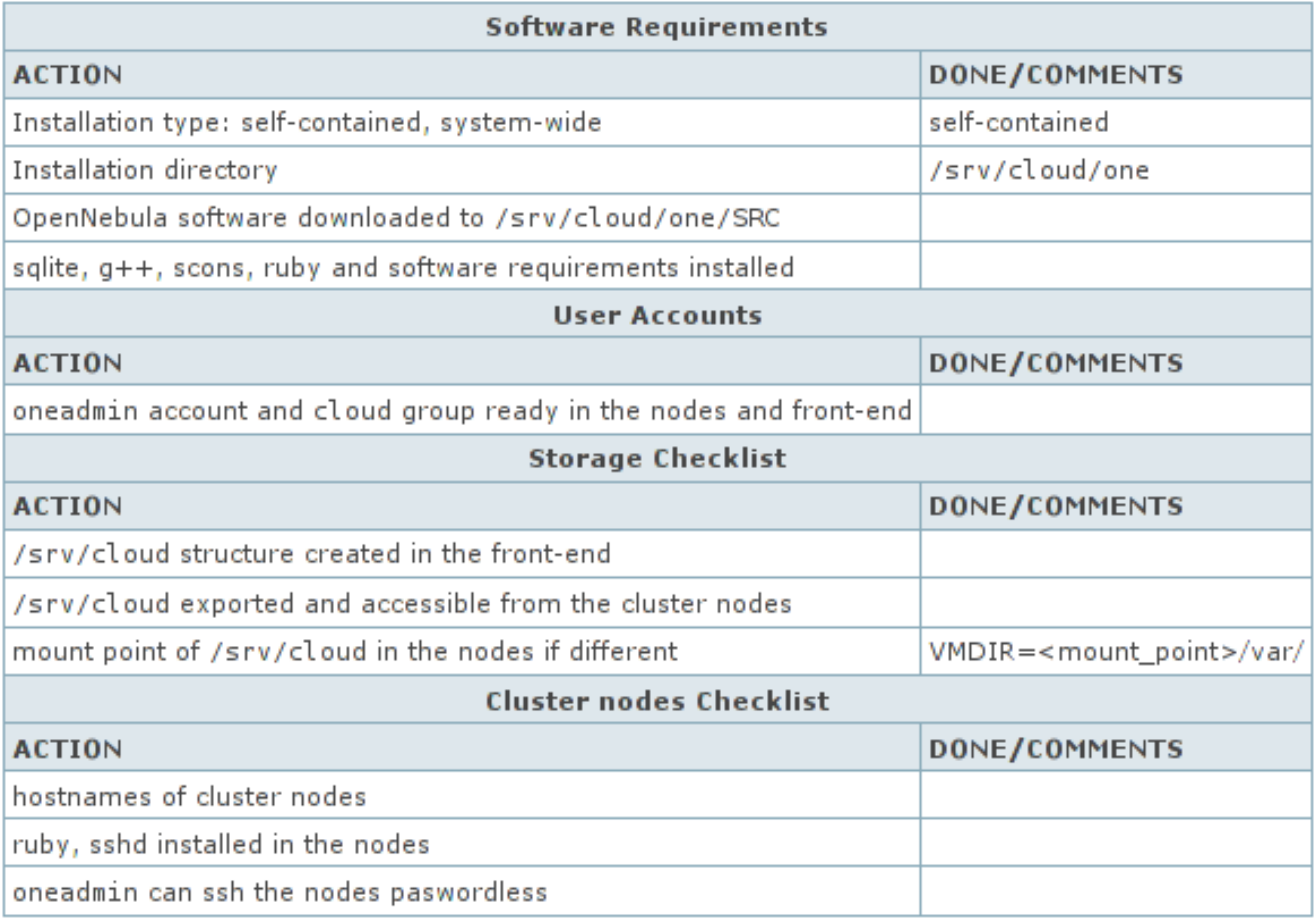
- Test the installation for the oneadmin account



```
no$ sudo /usr/sbin/xm list
Name       ID  Mem(MiB)  VCPUs  State    Time(s)
Domain-0    0       256      1  r------      8.2
```

- This ensures that oneadmin is capable of running VMs

Planning the Installation: Checklist



Software Requirements	
ACTION	**DONE/COMMENTS**
Installation type: self-contained, system-wide	self-contained
Installation directory	/srv/cloud/one
OpenNebula software downloaded to /srv/cloud/one/SRC	
sqlite, g++, scons, ruby and software requirements installed	
User Accounts	
ACTION	**DONE/COMMENTS**
oneadmin account and cloud group ready in the nodes and front-end	
Storage Checklist	
ACTION	**DONE/COMMENTS**
/srv/cloud structure created in the front-end	
/srv/cloud exported and accessible from the cluster nodes	
mount point of /srv/cloud in the nodes if different	VMDIR=<mount_point>/var/
Cluster nodes Checklist	
ACTION	**DONE/COMMENTS**
hostnames of cluster nodes	
ruby, sshd installed in the nodes	
oneadmin can ssh the nodes paswordless	

OpenNebula/Reservoir Training, January 27-28

Brussels, Belgium

Session 2
Administration and Basic Usage – Part I

Daniel Molina & Javier Fontán
dmolina/jfontan@opennebula.org

OpenNebula.org

RESERVOIR

SEVENTH FRAMEWORK PROGRAMME

European Commission Information Society and Media

Installing OpenNebula 2.0

- Grab the source code from /automount/share/reservoir/opennebula/2.0.1/opennebula-2.0.1.tar.gz and compile it!

```
fe~/SRC$ tar xzvf opennebula-2.0.1.tar.gz
fe~/SRC$ cd opennebula-2.0.1
fe~/SRC$ scons
```

- If there are problem with PKG_CONFIG_PATH:

```
fe~/SRC$ export PKG_CONFIG_PATH=/usr/lib/pkgconfig
```

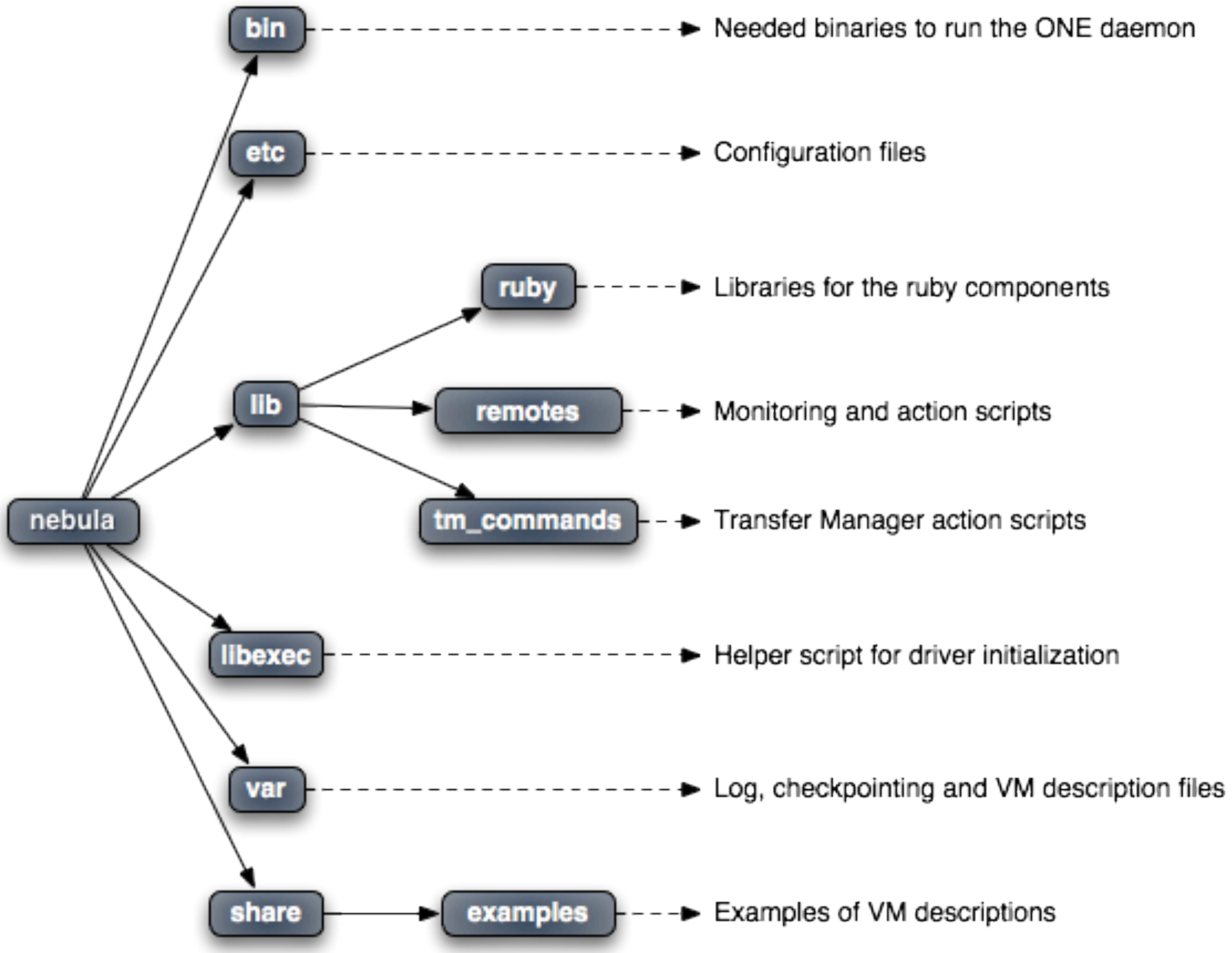
- Install the software in /srv/cloud/one (ONE_LOCATION)

```
fe$ export ONE_LOCATION=/srv/cloud/one/
fe$ ./install.sh -d $ONE_LOCATION

Check install.sh -h for other options
```

- Check and explore the installation tree

```
fe~$ ls -F
bin/  etc/  examples.desktop  include/  lib/  share/  SRC/  var/
```

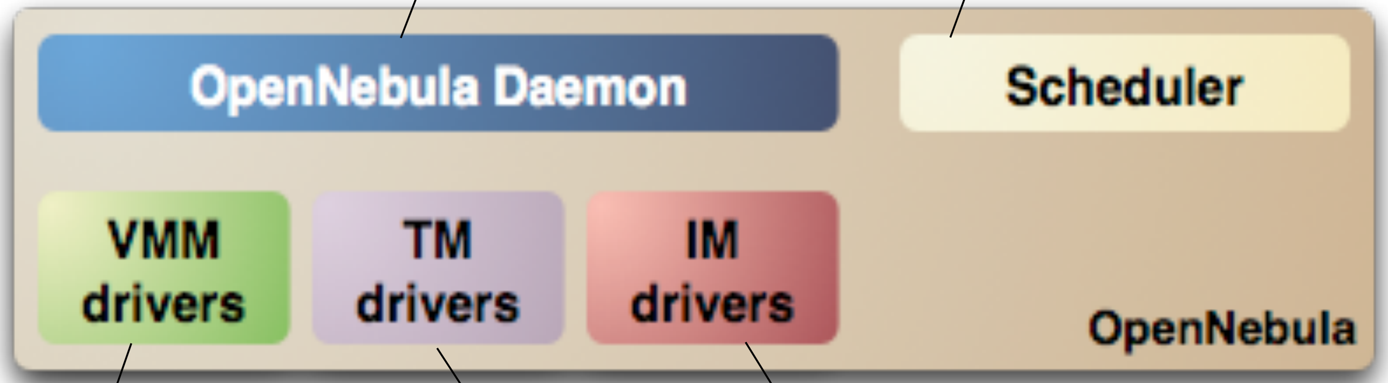
Installing OpenNebula 2.0


bin
Needed binaries to run the ONE daemon
etc
Configuration files
ruby
Libraries for the ruby components
lib
remotes
Monitoring and action scripts
nebula
tm_commands
Transfer Manager action scripts
libexec
Helper script for driver initialization
var
Log, checkpointing and VM description files
share
examples
Examples of VM descriptions

Configuring OpenNebula: The configuration interface

- `$ONE_LOCATION/etc/oned.conf`
- General configuration
- Defines the drivers used in the private cloud

- Match-making scheduler (default)
- Placement policies configured per VM



- `$ONE_LOCATION/etc/im_*/im_*.conf`
- Defines monitoring probes

- `$ONE_LOCATION/etc/vmm_*/vmm_*.conf`
- Defaults values for the hypervisor

- `$ONE_LOCATION/etc/tm_*/tm_*.conf`
- Defines action for generic storage operations

Configuring OpenNebula: The oned.conf file

- General configuration attributes
  - Monitoring intervals, HOST_MONITORING_INTERVAL VM_POLLING_INTERVAL
  - VM_DIR: Path to the VM directory for all the cluster nodes.
  - SCRIPTS_REMOTE_DIR: Remote path to store the monitoring and VM management script.
  - PORT : Port where oned will listen for xml-rpc calls
  - DB: Configuration attributes for the database backend
  - VNC_BASE_PORT: VNC ports are set to VNC_BASE_PORT + VMID
  - DEBUG_LEVEL

```
HOST_MONITORING_INTERVAL = 60
VM_POLLING_INTERVAL      = 60

#VM_DIR       = /srv/cloud/one/var

SCRIPTS_REMOTE_DIR  = /var/tmp/one
PORT                =  2633
DB = [ backend      = "sqlite" ]
VNC_BASE_PORT       = 5900
DEBUG_LEVEL         = 3
```

Configuring OpenNebula: The oned.conf file

- Physical Networks configuration
  - NETWORK_SIZE: default size for the virtual networks
  - MAC_PREFIX: Default prefix to be used in the auto-generated MAC addresses

```
NETWORK_SIZE = 254
MAC_PREFIX   = "02:00"
```

- Image Repository Configuration
  - IMAGE_REPOSITORY_PATH: by default $ONE_LOCATION/var/images
  - DEFAULT_IMAGE_TYPE: Can be: OS, CDROM, DATABLOCK
  - DEFAULT_DEVICE_PREFIX: hd, sd, xvd, vd

```
#IMAGE_REPOSITORY_PATH = /srv/cloud/var/images

DEFAULT_IMAGE_TYPE    = "OS"
DEFAULT_DEVICE_PREFIX = "hd"
```

Configuring OpenNebula: The oned.conf file

- Information Drivers, to monitor cluster nodes
  - name: identifies the driver
  - executable: absolute or relative to $ONE_LOCATION/lib/mads
  - arguments: a probe configuration file

```
IM_MAD = [
      name       = "im_xen",
      executable = "one_im_ssh",
      arguments  = ”xen" ]
```

- Transfer Drivers, to interface with the storage
  - name: identifies the driver
  - executable: path to driver executable
  - arguments: storage commands configuration file

```
TM_MAD = [
    name       = "tm_nfs",
    executable = "one_tm",
    arguments  = "tm_nfs/tm_nfs.conf" ]
```

Configuring OpenNebula: The oned.conf file

- Virtualization Drivers, to interface the hypervisors
  - name: identifies the driver
  - executable: absolute or relative to $ONE_LOCATION/lib/mads
  - arguments: (not needed for the distribution drivers)
  - default: default values for the hypervisor
  - type: format of the VM description file to be passed to the driver: xen, kvm or xml

```
VM_MAD = [
    name       = "vmm_xen",
    executable = "one_vmm_xen",
    arguments  = ”xen”,
    default    = "vmm_ssh/vmm_ssh_xen.conf",
    type       = "xen" ]
```

- Hooks, custom programs that are executed on specific events, e.g. VM creation.
- Hands on... Check and adjust the values of oned.conf for your cloud

Configuring OpenNebula: Accounts

- Accounts in OpenNebula
  - **oneadmin**, has enough privileges to perform any operation on any object. It is created the first time OpenNebula is started using the `ONE_AUTH` data
  - Regular **user accounts** must be created by `oneadmin` and they can only manage their own objects, or public ones.
- Configuring the oneadmin account
  - Environment variables: `ONE_AUTH`, `ONE_LOCATION` and `ONE_XMLRPC`

```
fe$ tail .bashrc
export ONE_LOCATION=/srv/cloud/one
export ONE_AUTH=$HOME/.one/one_auth
export PATH=$PATH:$ONE_LOCATION/bin
```

  - Create the password file

```
fe$ mkdir .one
fe$ cd .one
fe$ vi one_auth
oneadmin:onecloud
```

Configuring OpenNebula: Start & Stop

- Use the `one` script

```
fe$ source .bashrc
fe$ echo $ONE_AUTH
/srv/cloud/one/.one/one_auth

fe$ one start
oned and scheduler started

fe$ more $ONE_LOCATION/var/oned.log
Thu Jan 14 18:03:11 2010 [ONE][I]: Init OpenNebula Log system
Thu Jan 14 18:03:11 2010 [ONE][I]: Log Level: 3 [0=ERROR,1=WARNING,
2=INFO,3=DEBUG]
Thu Jan 14 18:03:11 2010 [ONE][I]: ------------------------------------
Thu Jan 14 18:03:11 2010 [ONE][I]:      OpenNebula Configuration File
Thu Jan 14 18:03:11 2010 [ONE][I]: ------------------------------------
```

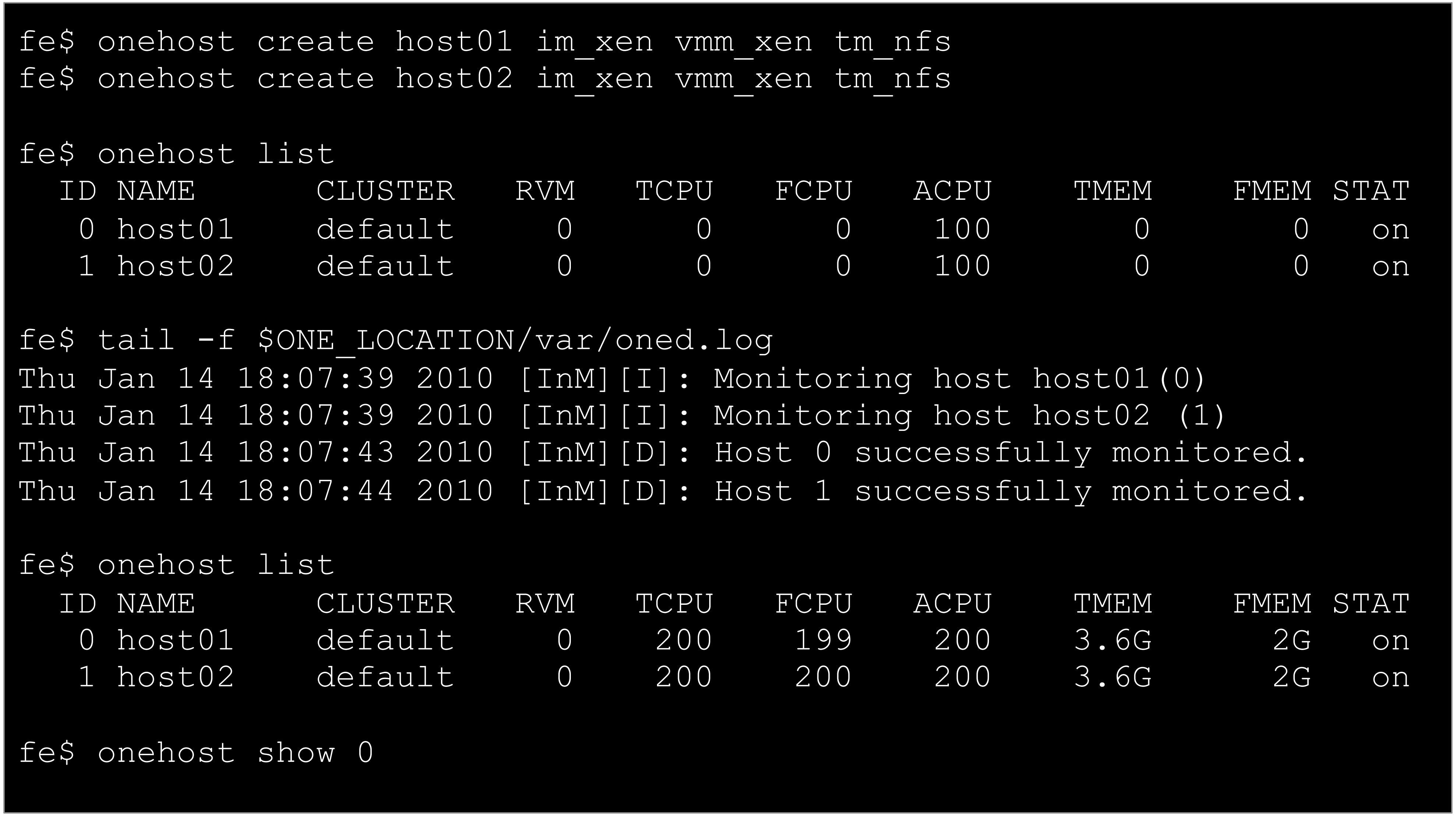
> Be sure to configure the oneadmin account (specially, create the ONE_AUTH file) before starting OpenNebula for the first time.

Configuring OpenNebula: Hosts & Clusters

- Hosts are defined with
  - *Hostname* of the node or IP
  - *Information Driver* to be used to monitor the host
  - *Storage Driver* to clone, delete, move or copy images into the host
  - *Virtualization Driver* to boot, stop, resume VMs in the host
- By default, all hosts belong to the *default* logical cluster. Clusters are managed using the **onecluster** command
  - Create & delete clusters
  - List the available clusters
  - Add & remove hosts from the clusters
- Hosts are managed with the **onehost** utility
  - Create & delete hosts
  - List the hosts
  - Show detailed information from a host
  - Enable/Disable a host

Configuring OpenNebula: Hosts

- Hands on... configure the hosts of your private cloud



```
fe$ onehost create host01 im_xen vmm_xen tm_nfs
fe$ onehost create host02 im_xen vmm_xen tm_nfs

fe$ onehost list
  ID NAME       CLUSTER    RVM    TCPU    FCPU    ACPU     TMEM     FMEM STAT
   0 host01     default      0       0       0     100        0        0   on
   1 host02     default      0       0       0     100        0        0   on

fe$ tail -f $ONE_LOCATION/var/oned.log
Thu Jan 14 18:07:39 2010 [InM][I]: Monitoring host host01(0)
Thu Jan 14 18:07:39 2010 [InM][I]: Monitoring host host02 (1)
Thu Jan 14 18:07:43 2010 [InM][D]: Host 0 successfully monitored.
Thu Jan 14 18:07:44 2010 [InM][D]: Host 1 successfully monitored.

fe$ onehost list
  ID NAME       CLUSTER    RVM    TCPU    FCPU    ACPU     TMEM     FMEM STAT
   0 host01     default      0     200     199     200     3.6G       2G   on
   1 host02     default      0     200     200     200     3.6G       2G   on

fe$ onehost show 0
```

Configuring OpenNebula: Clusters

- Hands on... configure the clusters of your private cloud

```
fe$ onecluster list
  ID      NAME
   0   default

fe$ onecluster create testing
fe$ onecluster addhost host02 testing

fe$ onehost list
  ID NAME       CLUSTER    RVM    TCPU    FCPU    ACPU     TMEM     FMEM STAT
   0 host01     default      0     200     184     184     3.6G       2G   on
   1 host02     testing      0     200     200     200     3.6G       2G   on

fe$ onecluster delete testing

fe$ onehost list
  ID NAME       CLUSTER    RVM    TCPU    FCPU    ACPU     TMEM     FMEM STAT
   0 host01     default      0     200     184     184     3.6G       2G   on
   1 host02     default      0     200     200     200     3.6G       2G   on
```

- Hands on... Explore and test the **onehost** and **onecluster** commands in your cloud

Configuring OpenNebula: Users

- Users are defined within OpenNebula by:
  - *ID* unique identifier for the user
  - *Name* of the user, used for authentication
  - *Password* used for authentication
- Users are managed with the oneuser utility
  - Create, list and delete users
  - Change users’ passwords
- Hands on... create new users in your private cloud and configure the “*user*” UNIX account

```
fe$ oneuser create helen mypass
User “Helen” should put helen:mypass in $ONE_AUTH or ~/.one/one_auth

fe$ oneuser list
 UID NAME       PASSWORD                                      ENABLE
   0 oneadmin   c24783ba96a35464632a624d9f829136edc0175e        True
   2 helen      34a91f713808846ade4a71577dc7963631ebae14        True

fe$ oneuser delete helen
```

Configuring OpenNebula: Log Files

- The operations of the OpenNebula daemon and scheduler are logged in:
  - oned: $ONE_LOCATION/var/oned.log, Its verbosity is set by DEBUG_LEVEL in $ONE_LOCATION/etc/oned.conf.
  - Scheduler (mm_sched): All the scheduler information is collected into the $ONE_LOCATION/var/sched.log file.
- VM logs and files are in $ONE_LOCATION/var/<VM_ID>, more in a few slides...
- Drivers can activate ONE_MAD_DEBUG in the associated RC file (or in $ONE_LOCATION/etc/defaultrc)

Using the Private Cloud: Virtual Networks

- A Virtual Network in OpenNebula
  - Defines a separated MAC/IP address space to be used by VMs
  - Each virtual network is associated with a physical network through a bridge
  - Virtual Networks can be isolated (at layer 2 level) with ebtables and hooks
- Virtual Network definition
  - **Name,** of the network
  - **Type**
    - **Fixed**, a set of IP/MAC leases
    - **Ranged,** defines a network range
  - **Bridge**, name of the physical bridge in the physical host where the VM should connect its network interface
  - **Public**: whether or not this Virtual Network can be used by VMs of any other user
- Virtual Networks are managed with the `onevnet` utility

Using the Private Cloud: Virtual Networks

- Hands on... explore the use of onevnet list, show, delete

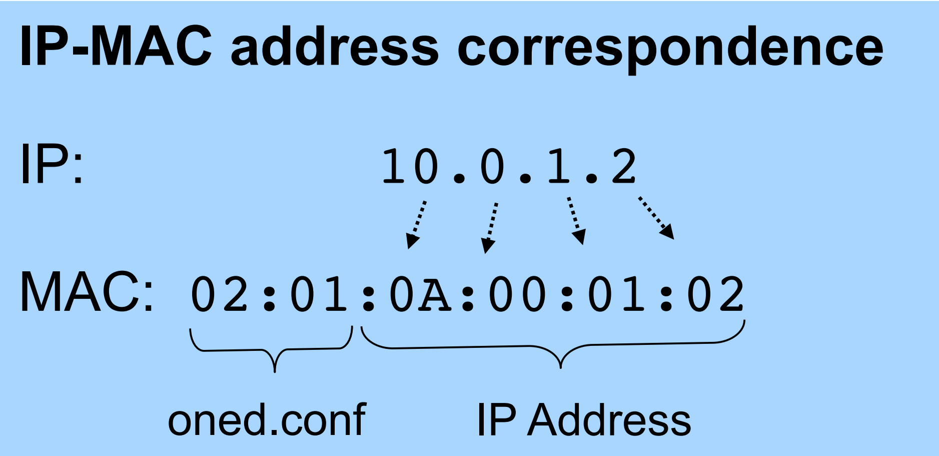
```
fe$ vi public.net
NAME    = ”Public"
TYPE    = FIXED
PUBLIC  = YES
BRIDGE  = xenbr0
LEASES  = [ IP=172.16.1.60+$CN  ]

fe$ vi onetd.net
NAME            = "One-TD"
TYPE            = RANGED
PUBLIC          = NO
BRIDGE          = xenbr0
NETWORK_SIZE    = 125
NETWORK_ADDRESS = 172.16.10+$CN.0

fe$ onevnet -v create public.net
fe$ onevnet -v create onetd.net
```

Using the Private Cloud: Virtual Networks

- Using a Virtual Network with your VMs
  - Define NICs attached to a given virtual network. The VM will get a NIC with a free MAC in the network and attached to the corresponding bridge

```
#A VM with two interfaces each one in a different vlan
NIC=[NETWORK=”Public”]
NIC=[NETWORK="One-TD"]

#Ask for a specific IP/MAC of the One-TD vlan
NIC=[NETWORK=”Public", IP=172.16.1.60+$CN ]
```

  - Prepare the VM to use the IP. Sample scripts to set the IP based on the MAC are provided for several Linux distributions.



IP-MAC address correspondence

IP: 10.0.1.2

MAC: 02:01:0A:00:01:02

oned.conf

IP Address

OpenNebula/Reservoir Training, January 27-28

Brussels, Belgium

Session 3
Administration and Basic Usage – Part II

Daniel Molina & Javier Fontán
dmolina/jfontan@opennebula.org

OpenNebula.org

SEVENTH FRAMEWORK PROGRAMME

European Commission
Information Society
and Media

Using the Private Cloud

- Preparing a VM to be used with OpenNebula
  - You can use any VM prepared for the target hypervisor
  - **Hint I**: Place the vmcontext.sh script in the boot process to make better use of vlans
  - **Hint II**: Do not pack useless information in the VM images:
    - swap. OpenNebula can create swap partitions on-the-fly in the target host
    - Scratch or volatile storage. OpenNebula can create plain FS on-the-fly in the target host
  - **Hint III:** Install once and deploy many; prepare master images
  - **Hint IV:** Do not put private information (e.g. ssh keys) in the master images, use the `CONTEXT`
  - **Hint V:** Pass arbitrary data to a master image using `CONTEXT`

Using the Private Cloud: ttylinux machine

- Hands on
  - Copy the ttylinux example from /automount/share/reservoir/opennebula/images/ttylinux-xen.tar

```
fe$ tar vzf ttylinux-xen.tar
```

Using the Private Cloud: Images

- An Image in OpenNebula's repository
  - Resource containing an operative system or data, to be used as a virtual machine disk.
  - This data can be saved overwriting the original image, or as a new OpenNebula image.
- Three different types of images
  - **OS**: contains a working operative system
  - **CDROM**: readonly data
  - **DATABLOCK**: A storage for data. Can be created either from previous existing data, or as an empty drive.
- Images are defined in an Image template
- Each Image has a unique name and ID in OpenNebula
- Once registered, Image files are stored in `$ONE_LOCATION/var/images`

Using the Private Cloud: Images

- Hands on... register a ttylinux OS image

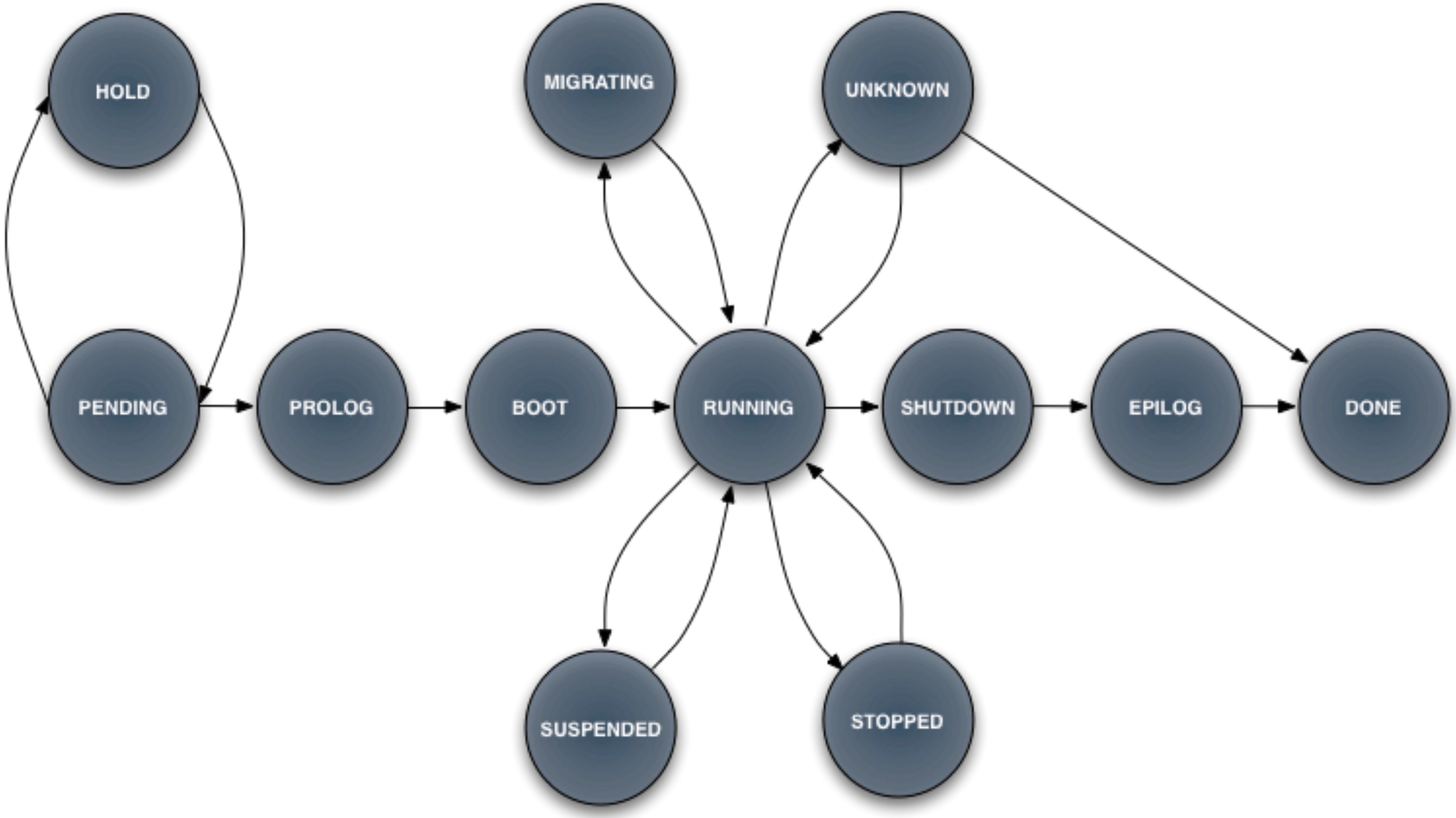
```
fe$ cat ttylinux-img.one
NAME          = "ttylinux"
TYPE          = OS
PATH          = /srv/cloud/one/ttylinux-xen/ttylinux.img
PUBLIC        = NO
PERSISTENT    = NO
DESCRIPTION   = "ttylinux OS"

fe$ oneimage register ttylinux-img.one
fe$ oneimage list
  ID     USER      NAME TYPE              REGTIME PUB PER STAT  #VMS
   0 oneadmin  ttylinux   OS   Dec 10, 2010 14:57  No  No  rdy     0

fe$ oneimage show 0
[ ... ]

fe$ tree /srv/cloud/one/var/images
/srv/cloud/one/var/images
`-- 8625d68b699fd30e64360471eb2c38fed47
```

Using the Private Cloud: Virtual Machines

- A Virtual Machine in OpenNebula
  - A **capacity** in terms memory and CPU
  - A set of **NICs** attached to one or more virtual networks
  - A set of **disk images**, to be "*transferred*" to/from the execution host.
  - A **state file** (optional) or recovery file, with the memory image of a running VM plus some hypervisor specific information.
- Virutal Machines are defined in a VM template
- Each VM has a unique ID in OpenNebula → the VM_ID
- All the files (logs, images, state files...) are stored in `$ONE_LOCATION/var/<VM_ID>`

Using the Private Cloud: Virtual Machines

- Virtual Machine Life-cycle



Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#---------------------------------------
# Name of the VM
#---------------------------------------
NAME = "vm-example" # Optional, Default: one-$VMID

#---------------------------------------
#               Capacity
#---------------------------------------
CPU    = "amount_of_requested_CPU"
MEMORY = "amount_of_requested_MEM"
VCPU   = "number of virtual cpus"

#---------------------------------------
#         OS and boot options
#---------------------------------------
OS = [
  kernel     = "path_to_os_kernel",     # para-virtualization
  initrd     = "path_to_initrd_image",  # para-virtualization
  kernel_cmd = "kernel_command_line",
  root       = "device to be mounted as root",
  bootloader = "path to the boot loader exec",
  boot       = "device to boot from" ]
```

Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#---------------------------------------
#      Features of the hypervisor
#---------------------------------------

FEATURES = [
  pae  = "yes|no",   # Optional, KVM
  acpi = "yes|no" ]  # Optional, KVM

#---------------------------------------
#          VM Disks, using Images
#---------------------------------------

DISK = [
  IMAGE     = "Name of the Image to use",
  IMAGE_ID  = ID,             # Optional, can be used instead of IMAGE
  BUS       = "ide, scsi, etc.",         # Optional
  TARGET    = "device_to_map_disk",      # Optional
  DRIVER    = "raw|qcow2|tap .. etc." ] # Optional
```

Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#---------------------------------------
#        VM Disks, advanced usage
#---------------------------------------

DISK = [
  type     = "floppy|disk|cdrom|swap|fs|block",
  source   = "path_to_disk_image_file|physical_dev",
  format   = "type for fs disks",
  size     = "size_in_GB",
  target   = "device_to_map_disk",
  bus      = "ide|scsi|virtio|xen",
  readonly = "yes|no",
  clone    = "yes|no",
  save     = "yes|no" ]
```

Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#---------------------------------------
#           Network Interfaces
#---------------------------------------

NIC = [
  network = "name_of_the_virtual_network",
  ip      = "ip_address",
  bridge  = "name_of_bridge_to_bind_if",
  target  = "device_name_to_map_if",
  mac     = "HW_address",
  script  = "path_to_script_to_bring_up_if",
  Model   = "NIC model"]

#---------------------------------------
#  I/O Interfaces
#---------------------------------------

INPUT = [
  type = "mouse|tablet",
  bus  = "usb|ps2|xen" ]
```

Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#---------------------------------------
#  I/O Interfaces
#---------------------------------------

GRAPHICS = [
  type   = "vnc|sdl",
  listen = "IP-to-listen-on",
  port   = "port_for_VNC_server",
  passwd = "password_for_VNC_server" ]

#---------------------------------------
#  Raw Hypervisor attributes
#---------------------------------------

RAW = [
  type = "xen|kvm",
  data = "raw_domain_configutarion"]
```

Not all the parameters are supported for each hypervisor. Complete reference and examples for all sections in

`http://opennebula.org/documentation:rel2.0:template`

Using the Private Cloud: Virtual Machines

- Hands on... define a new Virtual Machine:
  - Using the ttylinux Image
  - Connected to the Public and One-TD VirtualNetworks

```
fe$ cat ttylinux.one
NAME   = ttylinux-public
CPU    = 0.1
MEMORY = 64

DISK=[
  IMAGE=ttylinux,
  READONLY=no,
  TARGET=hda ]

NIC      = [ NETWORK=Public ]
NIC      = [ NETWORK=One-TD ]

FEATURES = [ ACPI=no ]

OS=[
  INITRD=/srv/cloud/one/ttylinux-xen/initrd.gz,
  KERNEL=/srv/cloud/one/ttylinux-xen/vmlinuz-xen,
  ROOT=hda1 ]
```

Using the Private Cloud: Virtual Machines

- Virtual Machines are managed with the onevm utility
  - Operations: create, deploy shutdown, livemigrate, stop, cancel, resume, suspend, delete, restart
  - Information: list, show, top, history

```
fe$ onevm create ttylinux.one

fe$ onevm list
   ID      USER      NAME STAT CPU      MEM          HOSTNAME           TIME
    0 oneadmin ttylinux pend   0      0K                   00 00:00:09

fe$ onevm show 0
[ ... ]

fe$ onevnet list
  ID USER     NAME                TYPE BRIDGE P #LEASES
   0 oneadmin One-TD           Ranged xenbr0 N       1
   1 oneadmin One-TD-Invisibl  Fixed xenbr0 N       0

fe$ oneimage list
  ID      USER      NAME TYPE              REGTIME PUB PER STAT  #VMS
   0 oneadmin  ttylinux   OS   Dec 10, 2010 14:57  No  No used     1

fe$ onevm top
```

Using the Private Cloud: Virtual Machines

- Hands on...
  - Create a basic VM
  - Create a couple of network enabled VMs
    - Check virtual network usage (onevnet)
  - Try control operations with the VMs
    - stop, shutdown, resume...
    - migrate – check xm list
  - Register a new persistent Datablock Image

```
NAME           = "storage"
TYPE           = DATABLOCK
PERSISTENT     = YES
SIZE           = 10
FSTYPE         = ext3
```

- Modify the template
  - Add one more NIC for the One-Td-Invisible network
  - Add another `DISK` for the persistent datablock image

OpenNebula/Reservoir Training, January 27-28

Brussels, Belgium

Session 4
Hybrid Cloud Computing

Daniel Molina & Javier Fontán
dmolina/jfontan@opennebula.org

OpenNebula.org

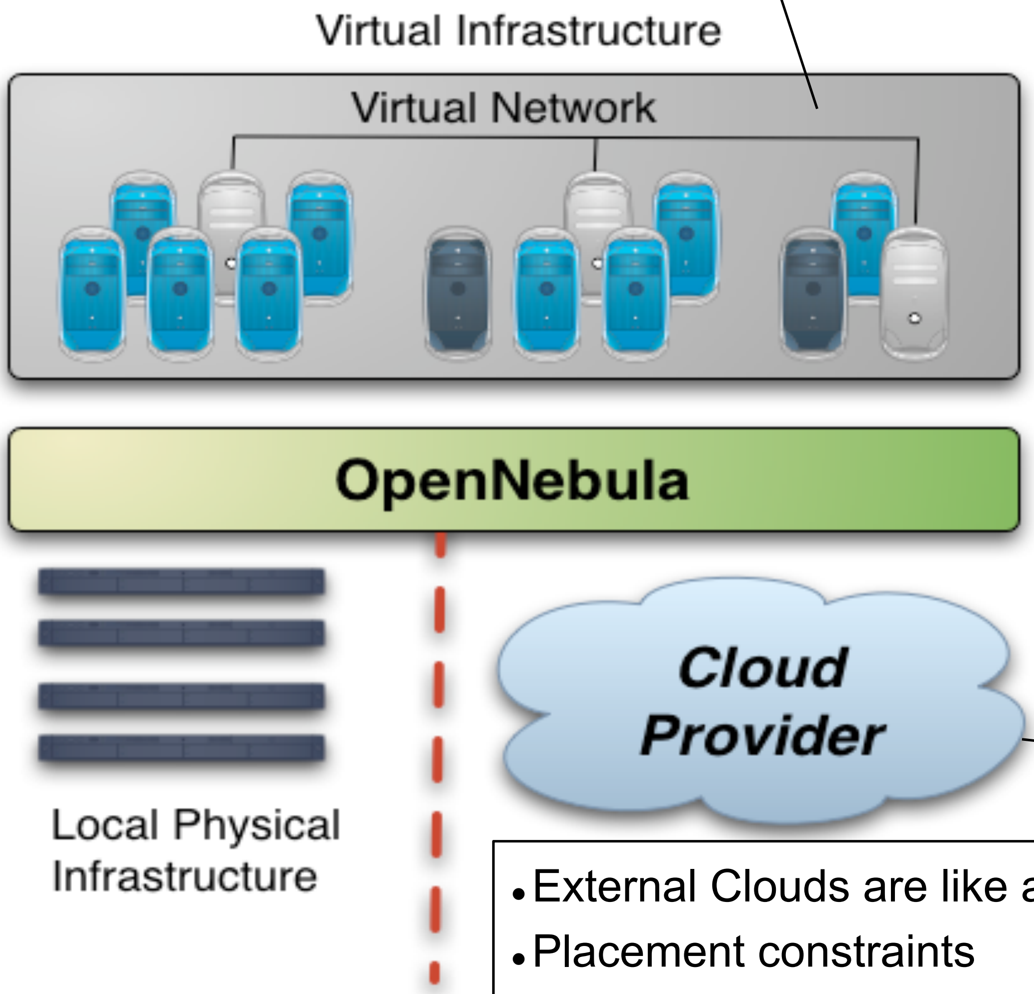
RESERVOIR

SEVENTH FRAMEWORK PROGRAMME

European Commission
Information Society
and Media

Hybrid Cloud Computing: Overview

- VMs can be local or remote
- VM connectivity has to be configured, usually VPNs



- External Clouds are like any other host
- Placement constraints
- OpenNebula distribution includes drivers to interact with Amazon EC2 and Elastic Hosts

Installing the Hybrid Cloud Components

Additional requirements:

- EC2 libraries and tools.
  - Grab the EC2 tools from /automount/share/reservoir/opennebula/ec2/tools

```
fe$ unzip ec2-api-tools.zip
fe$ cd ec2-api-tools-1.3-62308/

fe$ export EC2_HOME=`pwd`
fe$ export PATH=$EC2_HOME/bin:$PATH
```

- EC2 tools credentials:
  - Grab the EC2 credentials from /automount/share/reservoir/opennebula/ec2/certs

```
fe$ export EC2_PRIVATE_KEY=/srv/cloud/one/ec2/certs/pk.pem
fe$ export EC2_CERT=/srv/cloud/one/ec2/certs/cert.pem
```

Installing the Hybrid Cloud Components

- Hands on... try the EC2 tools (`ec2-*`)
  - `ec2-describe-images`: List and describe registered AMIs and AMIs you have launch permissions for.
  - `ec2-describe-instances`: List and describe your instances

```
$ ec2-describe-images
IMAGE    ami-0742a66e    /rubensm-amis.s3.amazonaws.com/
image.manifest.xml       418314910487     available        private
i386     machine
IMAGE    ami-e142a688    rubensm-amis.s3.amazonaws.com/
image.manifest.xml        418314910487     available        private
i386     machine
```

  - If you have problems with JAVA:

```
# yum install java-1.6.0-openjdk-devel-1.6.0.0-1.16.b17.el5
# export JAVA_HOME=/opt/jdk
```

Configuring the EC2 Hybrid Cloud Driver

- Hands on... Add the following drivers to oned.conf

```
IM_MAD = [
  name       = "im_ec2",
  executable = "one_im_ec2",
  arguments  = "im_ec2/im_ec2.conf" ] # No. of instances of each type

VM_MAD = [
  name       = "vmm_ec2",
  executable = "one_vmm_ec2",
  arguments  = "vmm_ec2/vmm_ec2.conf", # Defaults, e.g. keypair
  type       = "xml" ]

TM_MAD = [  #No actual transfers are made by OpenNebula to EC2
    name       = "tm_dummy",
    executable = "one_tm",
    arguments  = "tm_dummy/tm_dummy.conf" ]
```

Configuring the EC2 Hybrid Cloud Driver

- Hands on... Configure the account to be used with Amazon EC2

```
$ vim $ONE_LOCATION/etc/vmm_ec2/vmm_ec2rc
#----------------------------------------------------------------------
# EC2 API TOOLS Configuration.
#----------------------------------------------------------------------
EC2_HOME=/srv/cloud/one/ec2/tools
EC2_PRIVATE_KEY="/srv/cloud/one/ec2/certs/pk.pem"
EC2_CERT="/srv/cloud/one/ec2/certs/cert.pem"
```

- Hands on... You can limit the use of EC2 instances by modifying the IM file

```
$ vim $ONE_LOCATION/etc/im_ec2/im_ec2.conf
#----------------------------------------------------------------------
# Max number of instances that can be launched into EC2
#----------------------------------------------------------------------
SMALL_INSTANCES=5
LARGE_INSTANCES=
EXTRALARGE_INSTANCES=
```

Configuring the EC2 Hybrid Cloud Driver

- Amazon EC2 cloud is managed by OpenNebula as any other cluster node. Restart the oned, and check that the new drivers are loaded

```
$ one stop; one start
$ more $ONE_LOCATION/var/oned.log
Fri Jan 15 18:16:46 2010 [VMM][I]: Loading Virtual Machine Manager driv
Fri Jan 15 18:16:46 2010 [VMM][I]:      Loading driver: vmm_xen (XEN)
Fri Jan 15 18:16:47 2010 [VMM][I]:      Driver vmm_kvm loaded.
Fri Jan 15 18:16:47 2010 [VMM][I]:      Loading driver: vmm_ec2 (XML)
Fri Jan 15 00:16:47 2010 [InM][I]: Loading Information Manager drivers.
Fri Jan 15 00:16:47 2010 [InM][I]:      Loading driver: im_xen
Fri Jan 15 00:16:47 2010 [InM][I]:      Driver im_kvm loaded
Fri Jan 15 00:16:47 2010 [InM][I]:      Loading driver: im_ec2
```

- Hands on... Create your EC2 hybrid cloud by adding a new host

```
$ onehost create ec2 im_ec2 vmm_ec2 tm_dummy

$ onehost list
  ID NAME                       RVM    TCPU    FCPU    ACPU    TMEM    FMEM STAT
   0 host01                       0     200     200     200 2017004 1667080   on
   1 host02                       1     200     200     200 2017004 1681676   on
   2 ec2                          0     500     500     500 8912896 8912896   on
```

Configuring the EC2 Hybrid Cloud Driver

- You can use **several accounts** by adding a driver for each account (use the arguments attribute, -k and –c options). Then create a host that uses the driver

```
VM_MAD = [
  name        = "vmm_ec2_new",
  executable = "one_vmm_ec2",
  arguments  = "vmm_ec2/vmm_ec2.conf -k /srv/cloud/...",
  type        = "xml" ]
```

- You can use **multiple EC2 zones**, add a driver for each zone (use the arguments attribute, -u option). Then create a host that uses the driver

```
VM_MAD = [
  name        = "vmm_ec2_new",
  executable = "one_vmm_ec2",
  arguments  = "vmm_ec2/vmm_ec2.conf -u http://...",
  type        = "xml" ]
```

Using the EC2 Hybrid Cloud

- Virtual Machines can be instantiated locally or in EC2
  - The template must provide a description for both instantiation methods.
  - The EC2 counterpart of your VM (`AMI_ID`) must be available for the driver account
  - The EC2 VM template attribute:

```
EC2 = [
  AMI                = "ami_id for this VM",
  KEYPAIR            = "the keypair to use the instance",
  AUTHORIZED_PORTS = "ports to access the instance",
  INSTANCETYPE       = "m1.small...",
  ELASTICIP          = "the elastic ip for this instance",
  CLOUD              = "host (EC2 cloud) to use this description with"
]
```

Using the EC2 Hybrid Cloud

- Hands on... Add an EC2 counterpart to the ttylinux image

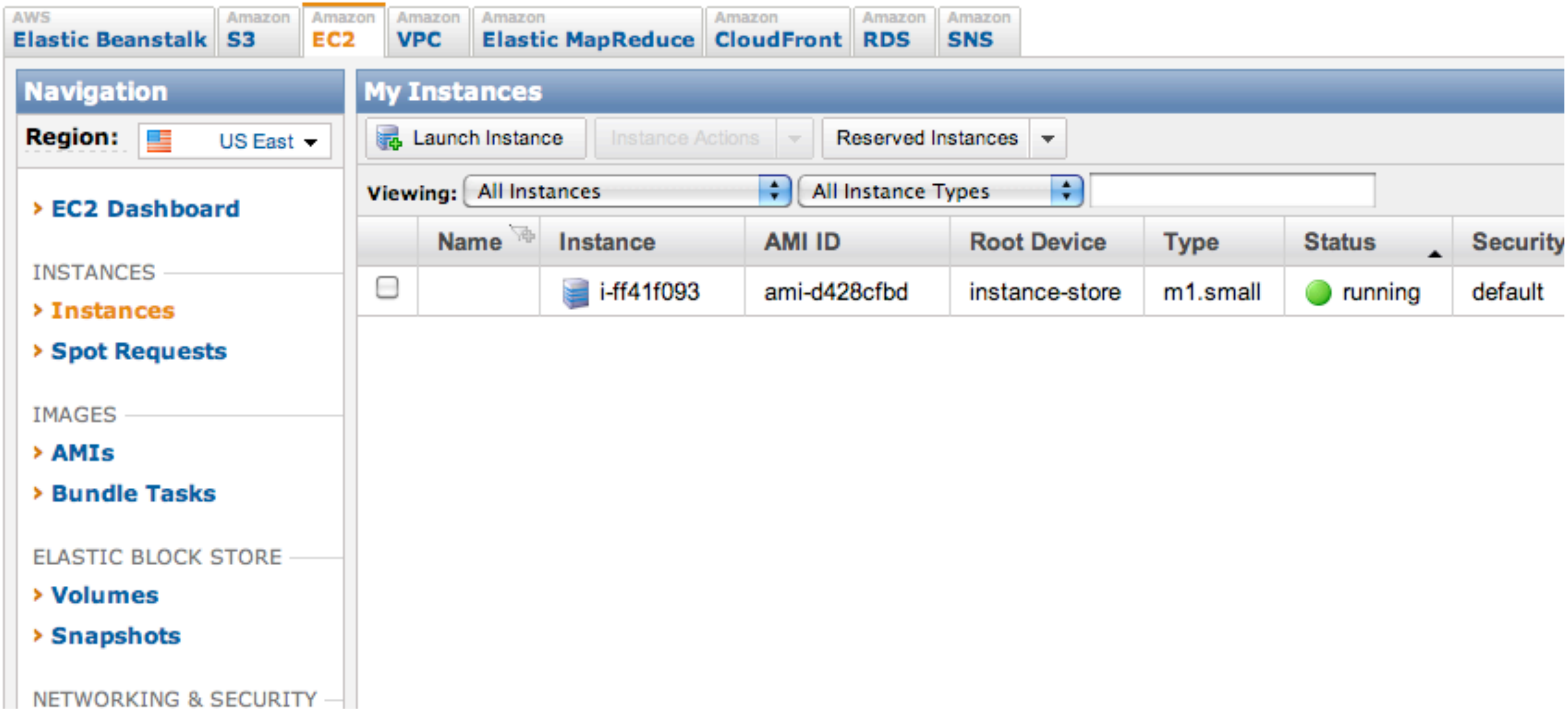
```
fe$ vi ttylinux.one
#EC2 template machine, this will be use wen submitting this VM to EC2
EC2 = [ AMI="ami-5e28d937",
        KEYPAIR=”td-keypair",
        AUTHORIZED_PORTS="22",
        INSTANCETYPE=m1.small]

#Add this if you want to use only EC2 cloud
REQUIREMENTS = "HOSTNAME = \"ec2\""
```

- Hands on... Create the VM and check progress

```
fe$ onevm create ttylinux.one
fe$ onevm list
  ID     USER     NAME STAT CPU     MEM        HOSTNAME        TIME
  16 oneadmin   one-16 runn   0       0             ec2 00 00:00:35
fe$ ec2-describe-instances
RESERVATION     r-5eff7536      418314910487    default
INSTANCE        i-bac3f0d2      ami-0572946c                            pending
keypair0        m1.small        2010-01-14T23:32:35+0000                us-
east-1a     aki-a71cf9ce     ari-a51cf9cc                   monitoring-
disabled
```

Using the EC2 Hybrid Cloud

- Hands on... Check the Amazon Web Service for the new Virtual Machine created through OpenNebula.
  - https://console.aws.amazon.com/ec2/



Region: US East

Viewing: All Instances | All Instance Types

	Name	Instance	AMI ID	Root Device	Type	Status	Security
☐		i-ff41f093	ami-d428cfbd	instance-store	m1.small	● running	default

Using the EC2 Hybrid Cloud

- Hands on... Log in the EC2 instance when running

```
fe$ onevm show 17
...
VIRTUAL MACHINE TEMPLATE
CPU=0.5
...
EC2=[
  AMI=ami-ccf615a5,
  AUTHORIZED_PORTS=22,
  INSTANCETYPE=m1.small,
  KEYPAIR=keypair ]
IP=ec2-72-44-62-194.compute-1.amazonaws.com
 ...
REQUIREMENTS=HOSTNAME = "ec2"
VMID=17

fe$ ssh -i keypair.pem root@ec2-72-44-62-194.compute-1.amazonaws.com
Linux ip-10-212-134-128 2.6.21.7-2.fc8xen-ec2-v1.0 #2 SMP Tue Sep 1
10:04:29 EDT 2009 i686
root@ip-10-212-134-128:~#

This costs money!
fe$ onevm shutdown 17
fe$ onehost disable ec2
fe$ onehost list
```

OpenNebula/Reservoir Training, January 27-28

Brussels, Belgium

Session 5
Public Cloud Computing

Daniel Molina & Javier Fontán
dmolina/jfontan@opennebula.org

OpenNebula.org

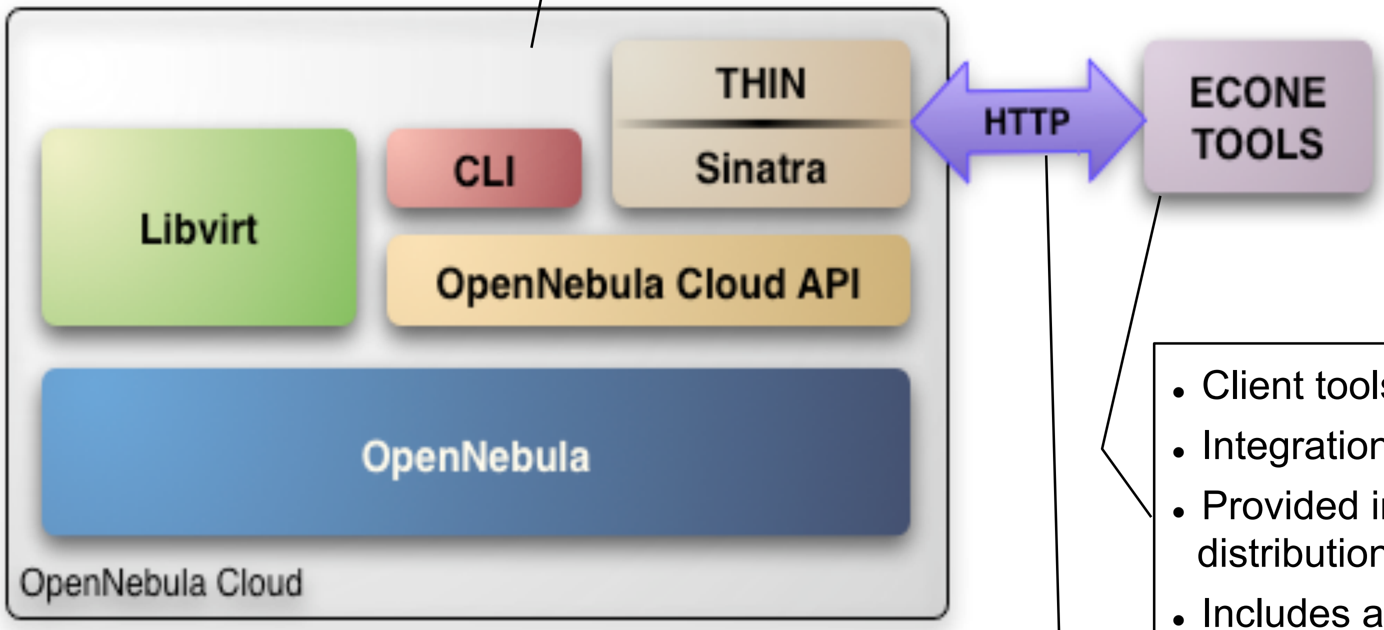
RESERVOIR

SEVENTH FRAMEWORK PROGRAMME

European Commission
Information Society
and Media

The Public Cloud: Overview

- You can use multiple interfaces for the Cloud
- Transparent to your setup:
  - Hypervisor
  - Storage Model
  - Hybrid configuration



- Client tools uses EC2 libraries
- Integration with EC2 tools
- Provided in the OpenNebula distribution
- Includes a simple S3 replacement

- Supports HTTP and HTTPS protocols
- *EC2 authentication* based on OpenNebula credentials
- Public Cloud users need an OpenNebula account

Installing the Public Cloud Components

- OpenNebula distribution supports two Cloud interfaces:
  - EC2 Query API
  - OCCI
- Additional requirements: EC2 development library, web server and web framework

```
fe# gem uninstall rack

fe# gem install rack --version ‘1.2.0’
fe# gem install sinatra
fe# gem install thin
fe# gem install amazon-ec2 --version ‘0.9.14’
fe# gem install uuid

Add a “FQDN” for our Public Cloud
fe# vim /etc/hosts
127.0.0.1       localhost
…
192.168.$CN.2   frontend cloud$CN.opennebula.org
```

Configuring the Public Cloud

- The EC2 service is configured in `$ONE_LOCATION/etc/econe.conf`
- Hands on... Study the configuration file and adjust it to your cloud

```
# OpenNebula sever contact information
ONE_XMLRPC=http://localhost:2633/RPC2

# Host and port where econe server will run
SERVER=cloud$CN.opennebula.org
PORT=4567

# SSL proxy that serves the API (set if is being used)
#SSL_SERVER=fqdm.of.the.server

# VM types allowed and its template file (inside templates directory)
VM_TYPE=[NAME=m1.small, TEMPLATE=m1.small.erb]
```

Configuring the Public Cloud

- You have to define the correspondence between types (simple) and local instantiation of VMs (hard, you should be fine by now)
  - Capacity allocated by this VM type (CPU, MEMORY)
  - Your cloud requirements, e.g. force to use a given kernel (OS) or place public VMs in a given set of cluster nodes (REQUIREMENTS)
  - The network used by Public VMs (NIC)

- VM Types are defined in `econe.conf`. Templates for the VM templates are in `$ONE_LOCATION/etc/ec2query_templates`

- Templates for VM Types are erb files <% Ruby code here %>, you should not need to modify that.

Configuring the Public Cloud

- Hands on... Prepare the m1.small type of your cloud to use ttylinux.one as a reference

```
$ more m1.small.erb
NAME    = eco-vm

#Adjust Capacity for this instance type
CPU     = 0.1
MEMORY = 64

OS      = [ kernel = /srv/cloud/one/ttylinux-xen/vmlinuz-xen,
            initrd = /srv/cloud/one/ttylinux-xen/initrd.gz,
            root   = hda1    ]

DISK    = [ IMAGE_ID = <%= erb_vm_info[:img_id] %> ]

NIC     = [ NETWORK =  "One-TD" ]

IMAGE_ID       = <%= erb_vm_info[:ec2_img_id] %>
INSTANCE_TYPE = <%= erb_vm_info[:instance_type ]%>

<% if erb_vm_info[:user_data] %>
CONTEXT = [
       EC2_USER_DATA="<%= erb_vm_info[:user_data] %>",
       TARGET="hdc” ]
<% end %>
```

Configuring the Public Cloud

- Hands on...
  - Create a new Public Cloud user

```
fe$ oneuser create ec2-user ec2-pass
fe$ oneuser list
  ID USER            PASSWORD
   0 oneuser         34c629abfcb47856b3d1c0a30798221aefb61605
   1 ec2-user        7030ddf34333388e9a7f0c13a6317ed4d66ac39f
```

  - Start the econe server

```
fe$ econe-server start

fe$ /usr/sbin/lsof -Pi

Check $ONE_LOCATION/var/econe-server.log for errors
```

Using the Public Cloud

- The econe-tools are a subset of the functionality provided by the onevm utility, and resembles the ec2-* cli
- Image related commands are:
  - `econe-upload`, place an image in the Cloud repo and returns ID
  - `econe-describe-images`, lists the images
  - `econe-register`, register an image
- Instance related commands are:
  - `econe-run-instances`, starts a VM using an image ID
  - `econe-describe-instances`, lists the VMs
  - `econe-terminate-instances`, shutdowns a VM
- User authentication is based in the OpenNebula credentials
  - `AWSAccessKeyId` is OpenNebula's username
  - `AWSSecretAccessKey` is OpenNebula's password

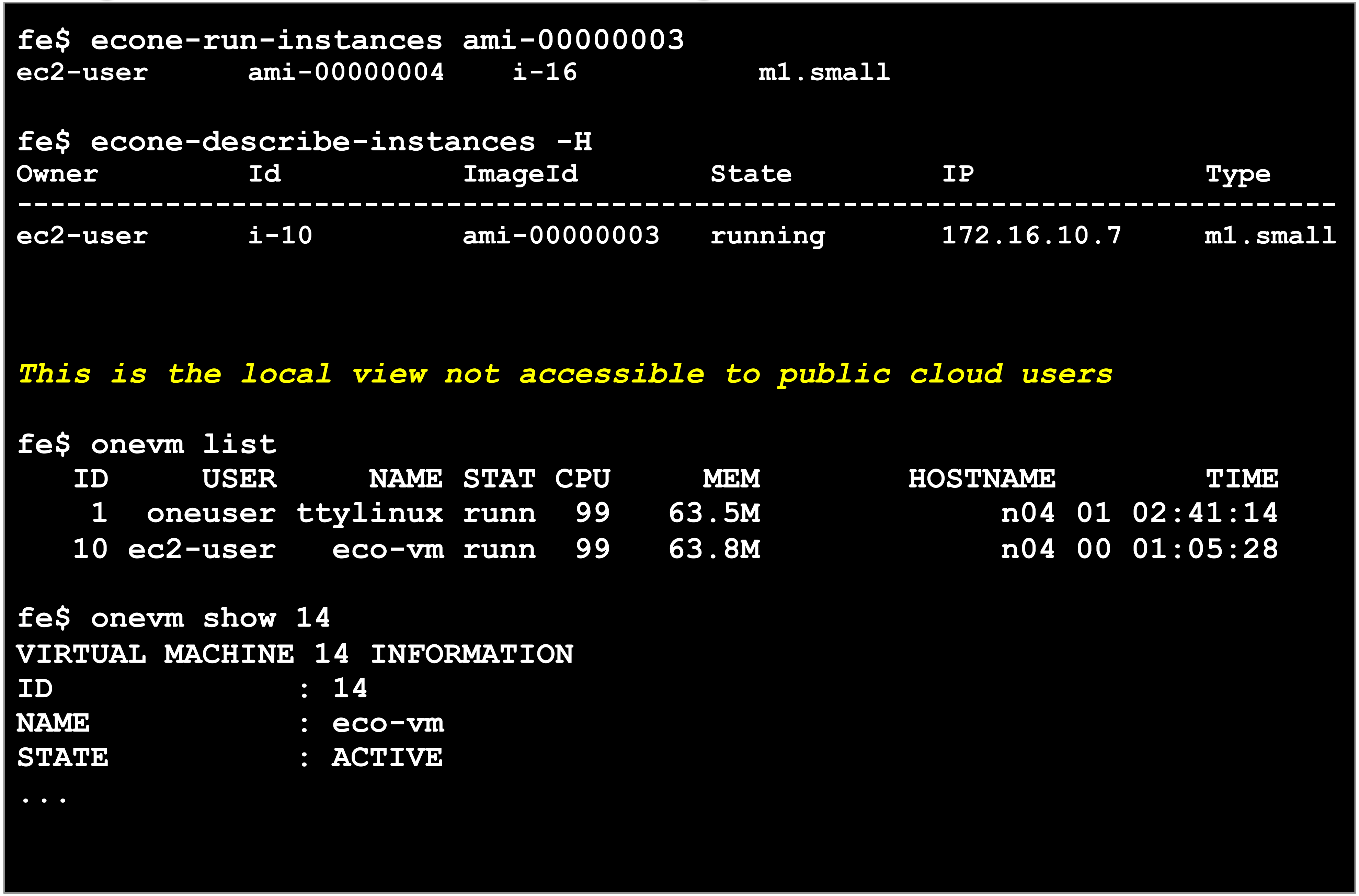
Using the Public Cloud

- Pass your credentials to the econe-tools by (in this order)
  - Command arguments (--access-key <username>,
    --secret-key <pass>)
  - Environment `EC2_ACCESS_KEY` and `EC2_SECRET_KEY`
  - Environment `ONE_AUTH`
- Point econe-tools to your target cloud
  - Command arguments (--url <http | https>://<fqdn>:<port>) port needed if not the default for the protocol
  - `EC2_URL` environment
- Hands on... upload the ttylinux image, and start it using the public cloud interface.
  - Compare the econe-* (public view) and one* (local view) evolution and information
  - Check the template build by the econe server (onevm show)

Using the Public Cloud, uploading an Image

```
fe$ econe-upload -U http://node-x.opennebula.org:4567 --access-key ec2-
user --secret-key ec2-pass /srv/cloud/images/ttylinux/ttylinux.img
Success: ImageId ami-00000003

fe$ export EC2_URL=http://cloud$CN.opennebula.org:4567
fe$ export EC2_ACCESS_KEY=ec2-user
fe$ export EC2_SECRET_KEY=ec2-pass

fe$ econe-describe-images -H
Owner        ImageId       Status         Visibility   Location
-----------------------------------------------------------------------
ec2-user     ami-00000003  available      private      23151fac850e5...


This is the local view not accessible to public cloud users
fe$ oneimage list
  ID     USER                 NAME TYPE              REGTIME PUB PER STAT  #VMS
   0  oneuser             ttylinux   OS   Jan 21, 2011 13:59  No  No used     1
   1  oneuser              storage   DB   Jan 21, 2011 13:59  No Yes  rdy     0
   3 ec2-user ec2-71654e30-0872-01   OS   Jan 22, 2011 16:27  No  No  rdy     0

$ oneimage show 3
IMAGE  INFORMATION
ID             : 3
...
```

Using the Public Cloud, running an Instance



```
fe$ econe-run-instances ami-00000003
ec2-user     ami-00000004    i-16          m1.small

fe$ econe-describe-instances -H
Owner        Id           ImageId        State        IP            Type
------------------------------------------------------------------------------
ec2-user     i-10         ami-00000003   running      172.16.10.7   m1.small



This is the local view not accessible to public cloud users

fe$ onevm list
   ID     USER     NAME STAT CPU     MEM        HOSTNAME        TIME
    1  oneuser ttylinux runn  99   63.5M             n04 01 02:41:14
   10 ec2-user   eco-vm runn  99   63.8M             n04 00 01:05:28

fe$ onevm show 14
VIRTUAL MACHINE 14 INFORMATION
ID               : 14
NAME             : eco-vm
STATE            : ACTIVE
...
```

Configuring SSL access for the Public Cloud

- SSL security is handle by a proxy that forwards the request to the EC2 Query Service and takes back the answer to the client
- Requirements:
  - A server certificate for the SSL connections
  - An HTTP proxy that understands SSL
  - EC2Query Service configuration to accept petitions from the proxy
- Hands on... Install the proxy (lighttpd) and get the certificates for your cloud

```
fe# yum install lighttpd

fe# cp /automount/share/reservoir/opennebula/certs/server.pem /etc/
lighttpd/server.pem
```

Configuring SSL access for the Public Cloud

- Hands on... configure the lighttpd proxy

```
# vim /etc/lighttpd/lighttpd.conf
server.modules              = (
            "mod_access",
            "mod_alias",
            "mod_accesslog",
            "mod_compress",
            "mod_proxy"
...
## bind to port (default: 80)
server.port                = 443
...
#### proxy module
proxy.server                = ( "" =>
                                ("" =>
                                 (
                                   "host" => "127.0.0.1",
                                   "port" => 4567
                                 )
                                 )
                             )
#### SSL engine
ssl.engine                  = "enable"
ssl.pemfile                 = "/etc/lighttpd/server.pem"
```

Configuring SSL access for the Public Cloud

- Hands on... configure the econe server

```
$ vim /srv/cloud/one/etc/econe.conf

#SERVER=node-15.opennebula.org
SERVER=127.0.0.1
PORT=4567

# SSL proxy that serves the API (set if is being used)
SSL_SERVER=cloud$CN.opennebula.org
```

- Hands on... restart services (lighttpd and econe-server) and try your new SSL cloud access (https://node-x.opennebula.org:443)

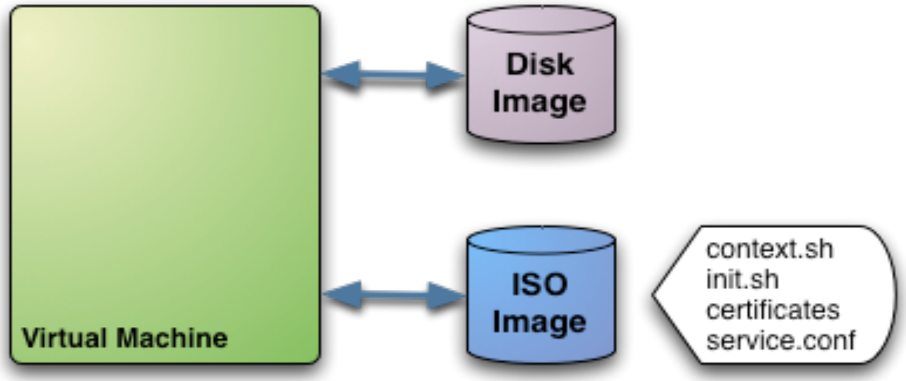
OpenNebula Technology Days, 20-21 July 2010

Session 6
Advanced Usage

Javier Fontán
jfontan@fdi.ucm.es

Using the Private Cloud: Virtual Machines

- Context contains data to be passed to the VM at boot time



```
#-----------------------------------
#  Context for the VM
#    values can be:
#    $<template_variable>
#    $<template_variable>[<attribute>]
#    $<template_variable>[<attribute>, <attribute2>=<value2>]
#    $<vm_id>.<context_var>
#-----------------------------------

CONTEXT = [
  var_1 = "value_1",#Will be in context.sh as var_1=”val_1” (sh syntax)
  var_n = "value_n",#Will be in context.sh as var_N=”val_N” (sh syntax)
  files = "space-separated list of paths to include in context device",
  target= "device to attach the context device" ]
```

Using the Private Cloud: Virtual Machines

- Hands on... Add custom ssh keys the VM image
  - Check boot process of the ttylinux VM (systemrc.local) it will
    - mount iso (do it yourself and see the ISO layout...)
    - Source context.sh
    - In this example it will execute init.sh so you can try anything

```
CONTEXT = [
    files      = "/srv/cloud/one/ttylinux-xen/init.sh /srv/cloud/
one/.ssh/id_rsa.pub",
    target      = "hdc",
    root_pubkey = "id_rsa.pub"
]
```

```
$ more init.sh
#!/bin/bash
if [ -f /mnt/context/context.sh ]
then
  . /mnt/context/context.sh
fi
if [ -f /mnt/context/$ROOT_PUBKEY ]; then
        cat /mnt/context/$ROOT_PUBKEY >> /root/.ssh/authorized_keys
fi
```

Using the Private Cloud: Virtual Machines

- Tunning the placement of VMs with the Match-making scheduler
  - First those hosts that do not meet the VM requirements are filtered out (REQUIREMENTS)
  - RANK is evaluated for the remaining hosts
  - That with the highest RANK is used for the VM
- Placement policies are specified per VM

```
#---------------------------------------
#             Scheduler
#---------------------------------------
# Use Host Monitor attributes
REQUIREMENTS = "Bool_expression_for_reqs"
RANK         = "Arith_expression_to_rank_hosts"
```

- Hands on... try a simple VM pinning

```
REQUIREMENTS = "HOSTNAME=\”...\”"
```

- Hands on... try a simple load-aware policy

```
RANK = FREECPU
```

Customizing and Extending your Cloud

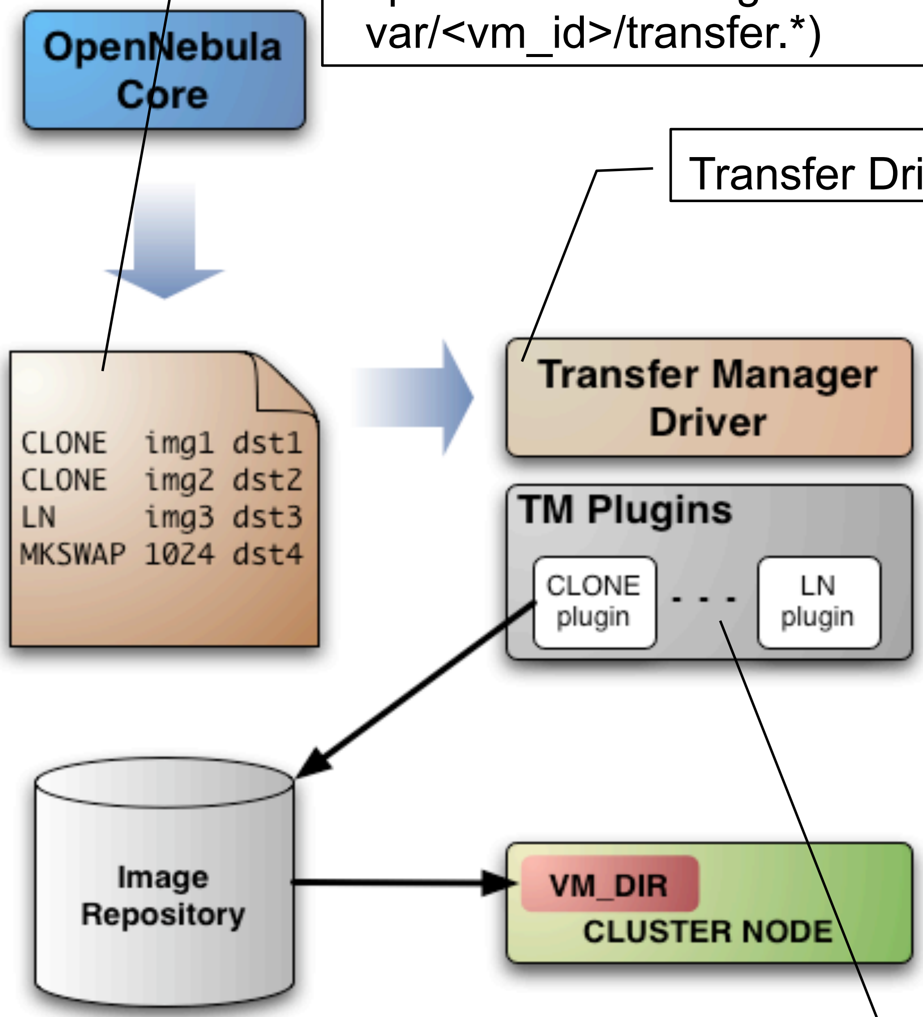
- You can customize your cloud by:
  - Tunning or adapting the transfer operations to your **storage back-end**
  - Adding new **monitorization** probes to improve the VM placement
  - Adjusting VM operations to your hypervisor installation
  - Trigger **custom actions** on specific VM events (e.g. “on VM creation update the accounting DB” or “on VM shutdown send an email”)
- You can extend your cloud by:
  - Developing new drivers for other hypervisors
  - Developing new drivers for other storage back-ends
  - Developing Cloud applications using the OpenNebula API or the Cloud APIs

> OpenNebula is very scripting friendly, drivers can be written in any language. You can modify the current ones or use them as templates for new ones.

Customizing the Storage of your Cloud

OpenNebula issue generic storage operations (check var/<vm_id>/transfer.*)

Transfer Driver process the core script



```
CLONE  img1 dst1
CLONE  img2 dst2
LN     img3 dst3
MKSWAP 1024 dst4
```

Action programs interpret the semantics of generic actions depending on the storage back.-end

Customizing the Storage of your Cloud

- OpenNebula requests the following abstract operations over a VM image
  - `CLONE`: This action will basically make a copy of the image from ORIGIN to DESTINATION.
  - `LN`: Creates a symbolic link in DESTINATION that points to ORIGIN
  - `MKSWAP`: Generates a swap image in DESTINATION. The size is given in ORIGIN in MB.
  - `MKIMAGE`: Creates a disk image in DESTINATION and populates it with the files inside ORIGIN directory.
  - `DELETE`: Deletes ORIGIN file or directory.
  - `MV`: Moves ORIGIN to DESTINATION.

Customizing the Storage of your Cloud

- Actions are defined in

`$ONE_LOCATION/etc/tm_<storage>/tm_<storage>.conf`

```
$ more /srv/cloud/one/etc/tm_ssh/tm_ssh.conf
CLONE   = ssh/tm_clone.sh
LN      = ssh/tm_ln.sh
MKSWAP  = ssh/tm_mkswap.sh
MKIMAGE = ssh/tm_mkimage.sh
DELETE  = ssh/tm_delete.sh
MV      = ssh/tm_mv.sh
```

- Actions scripts are placed in

`$ONE_LOCATION/lib/tm_commands/<storage>/`

```
$ ls /srv/cloud/one/lib/tm_commands/ssh/
tm_clone.sh    tm_delete.sh  tm_mkimage.sh  tm_mv.sh
tm_context.sh  tm_ln.sh      tm_mkswap.sh
```

Customizing the Storage of your Cloud

- Hands on... Take a look to the tm_clone.ssh

```
. $TMCOMMON
...
log "Creating directory $DST_DIR"
exec_and_log "ssh $DST_HOST mkdir -p $DST_DIR"
...
case $SRC in
http://*)
    log "Downloading $SRC"
    exec_and_log "ssh $DST_HOST wget -O $DST_PATH $SRC"
    ;;

*)
    log "Cloning $SRC"
    exec_and_log "scp $SRC $DST"
    ;;
esac

exec_and_log "ssh $DST_HOST chmod a+w $DST_PATH"
```

- Hands on... Check the semantics of other operations for the ssh storage, e.g. tm_ln.ssh

Storage Customization Examples

- Make swap images local to the physical node executing the VM
  - The script that generates swap images is called MKSWAP
  - Swap images are usually generated in VM directory
  - Link the newly create swap image to the VM directory
- Make OpenNebula aware of compressed images
  - Images are cloned by CLONE script

Customizing the Information System

- OpenNebula gets host information by executing an arbitrary number of probes
- A probe is a program that returns the monitorization metric in the form

```
METRIC_NAME = VALUE
```

- Probes are configured in

```
$ONE_LOCATION/etc/im_<hypervisor>/im_<hypervisor>.conf
```

And placed in

```
$ONE_LOCATION/lib/im_probes
```

- Probe information is mainly used for VM placement

Customizing the Information System

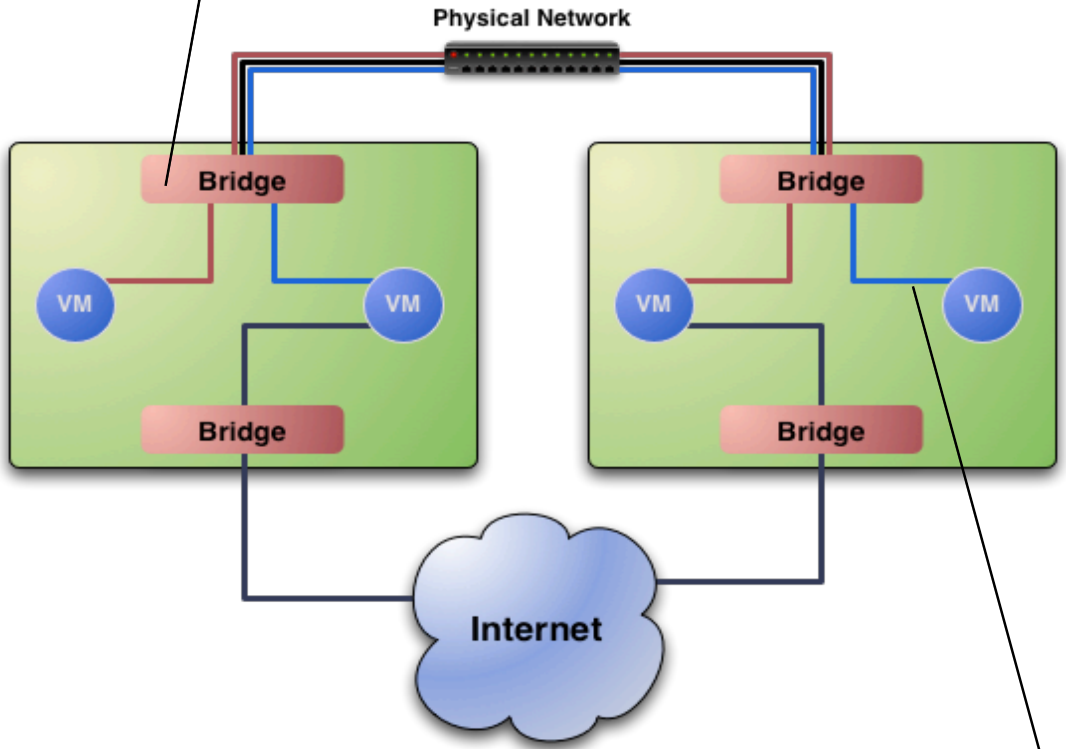
- Hands on... Take a look to the default probes defined for KVM

```
$ more /home/ruben/Virtual/one/etc/im_kvm/im_kvm.conf
cpuarchitecture=architecture.sh
nodename=name.sh
cpu=cpu.sh
kvm=kvm.rb

$ more name.sh
#!/bin/sh

echo HOSTNAME=`uname -n`
```

- Hands on... Create a new probe that returns the number of VMs in `RUNNING_VMS` (e.g. you can use virsh, pgrep kvm...). Use the new metric to pack VMs (`RANK=RUNNING_VMS`).

Customization with Hooks: Network Isolation

- IN: Only Ethernet frames from a MAC in Red LAN
- OUT: Only Ethernet frames from the MAC assigned by OpenNebula



- Networks are isolated at layer 2
- You can put any TCP/IP service as part of the VMs (e.g. DHCP, nagios...)

Customization with Hooks: Network Isolation

- Requirements (this has to be done in all the cluster nodes)
  - Check that ebtables package is installed
  - Allow oneadmin to use the ebtables command through sudo

```
#visudo
...
oneadmin    ALL=(ALL) NOPASSWD: /sbin/ebtables *
...
```

- Configure the hooks for OpenNebula

```
VM_HOOK = [
    name      = "ebtables-start",
    on        = "running",
    command   = "/srv/cloud/one/share/hooks/ebtables-kvm",
    arguments = "one-$VMID",
    remote    = "yes" ]
VM_HOOK = [
    name      = "ebtables-flush",
    on        = "done",
    command   = "/srv/cloud/one/share/hooks/ebtables-flush",
    arguments = "",
    remote    = "yes" ]
```

Customization with Hooks: Network Isolation

- Hands on... Start a couple of VMs in Networks Red and Blue.
  - Check the ebtables rules in the hosts
  - Check connectivity between VMs
  - Change the network mask of the VMs and check connectivity
  - Shutdown and check the ebtables rules